Common Gateway Services CSCI

Thor DP3

Version 2.1
November 18, 1997

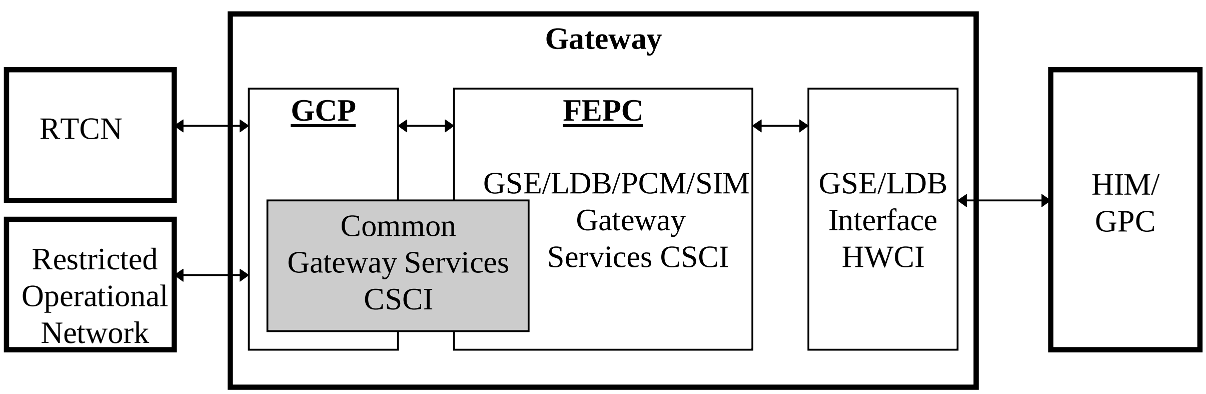
Error! Bookmark not defined.

1. Common Gateway Services CSCI

1.1 Common Gateway Services CSCI Introduction

1.1.1 Common Gateway Services CSCI Overview

The Common Gateway Services CSCI provides the Gateway functions that are common among all Gateway types. It is software resident on both the Gateway Control Processor (GCP) and the Front End Process Controller (FEPC). The Common Gateway Services CSCI's primary services are to provide and manage network interfaces for the Gateway, provide common command routing and processing functions, and provide Gateway wide subsystem integrity.



1.1.2 Common Gateway Services CSCI Operational Description

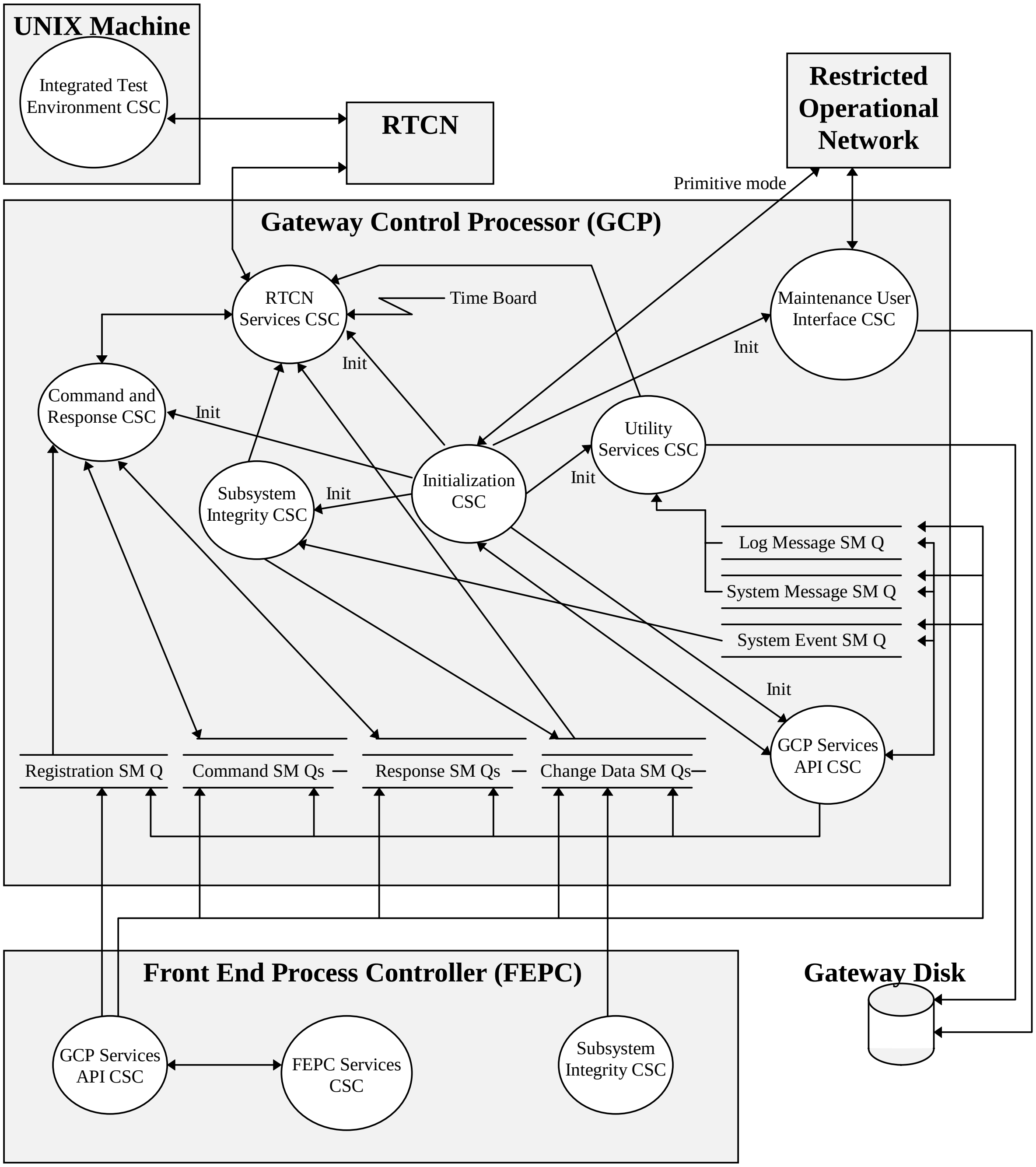
The Common Gateway Services CSCI is initiated by the Real Time Operating System (RTOS) resident on the Gateway's local disk. Initially it will spawn all the necessary tasks to support the Gateway services. All network interfaces for the Gateway are also provided by the Common Gateway Services CSCI.

During initialization, all Common Gateway Services CSCs are spawned by the Gateway Initialization CSC. Each Gateway resident processor registers its initialization parameters using the Registration Shared Memory Message Queue and the Gateway GCP Services API CSC. The Gateway FEPC Services CSC contains a generic set of initialization, and command and measurement processing routines which are applicable across unique FEPC types. These routines automate the initialization and command and measurement processing functions. The Gateway Initialization CSC also registers the initialization parameters for the GCP. In the case of a failure during Boot or SCID Initialization, the Gateway will enter the Primitive mode and will provide minimal operating system functionality over the Restricted Operational Network.

All Commands incoming to the Gateway are received asynchronously over the RTCN using the Gateway RTCN Services CSC. Commands are passed to the Gateway Command and Response CSC and then forwarded to the correct Gateway processor via the Command shared memory message queues. The Gateway resident processors read the commands out of the queues using the Gateway GCP Services API CSC. The Gateway resident processor may also generate responses to the issuers of the commands using the Gateway GCP Services API CSC. This places the outgoing response in the response queues where it will be read by the Gateway Command and Response CSC and forwarded to the Gateway RTCN Services CSC for transmission over the RTCN.

Change Data is generated by the Gateway resident processor and is placed on the Change Data queues using the Gateway GCP Services API CSC. The Gateway RTCN Services CSC is then responsible for reading Change Data entries off the queue, building a Change Data packet, and sending it over the RTCN at the System Synchronous Rate. The System Synchronous Rate will be provided as a software interrupt by the Gateway's local time board.

Gateway Health and Status will be accomplished by the Gateway Subsystem Integrity CSC. This CSC is responsible for task monitoring, and tracking processor health counts. All health and status information is transmitted as Function Designator entries in Change Data packets. The Change Data path of communication described above is used as the medium of their transmission. The Gateway Subsystem Integrity CSC is also responsible for the processing and generation of System Event codes.



The Gateway Utility Services CSC provides several generic capabilities to all resources in the Gateway. Error/message logging, and System Messages services are provided. This CSC also contains a Recovery Dump facility which will log all task, variable, and system information to the Shuttle Data Center and the Gateway disk if a fatal error occurs in the Gateway. The Gateway Utility Services CSC also provides a developmental means of storing a detailed error database for deciphering errors and other messages that may come to light in an operational environment. This database will be accessible using the Gateway Maintenance User Interface CSC.

The Maintenance User Interface CSC provides a network server for access to the Gateway over the Restricted Operational Network. This interface provides extensive CSC monitoring and Gateway Disk inspection capabilities. The capabilities to access verbose error messages, and decode Recovery Dumps are all accomplished by this Interface. It also provides limited internal commanding capabilities.

The Gateway Integrated Test Environment CSC provides a means of emulating the CCP, DDP, and Ops CM server. It operates over the RTCN on any machine that is not a Gateway. Command and Control and System Control commands may be sent using this CSC. It also provides an RTCN Analyzer that dumps information on a specific RTCN data stream.

1.2 Common Gateway Services Computer Software Components (CSCs)

Common Gateway Services CSCI is composed of the following CSCs:

- Gateway Initialization CSC
- Gateway FEPC Services CSC
- Gateway Command and Response CSC
- Gateway RTCN Services CSC
- Gateway Subsystem Integrity CSC
- Gateway Utility Services CSC
- GCP Services API CSC
- Gateway Maintenance User Interface CSC
- Gateway Integrated Test Environment CSC

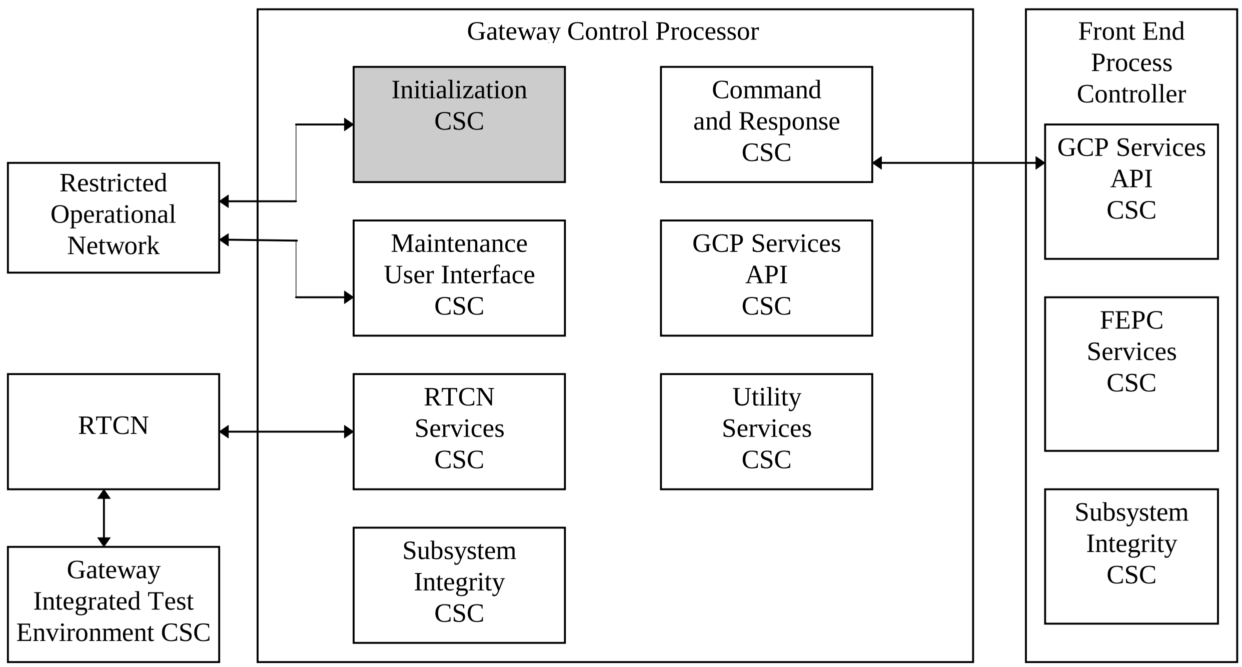
2. Gateway Initialization CSC

2.1 Gateway Initialization CSC Introduction

2.1.1 Gateway Initialization CSC Overview

The Gateway Initialization CSC is responsible for the initialization sequence of the Gateway, and for the initialization and termination of all other Gateway CSCs. It is part of the Common Gateway Services CSCI and is resident in the GCP.

Common Gateway Services CSCI



2.1.2 Gateway Initialization CSC Operational Description

The Gateway Initialization CSC controls the initialization sequence of the Gateway. The Gateway's initialization sequence is divided into five Initialization modes that allow a synchronized boot sequence between all the Single Board Computers resident in the Gateway. Gateway Initialization CSC also supports a Primitive mode that allows recovery from a Boot or SCID Initialization error.

2.2 Gateway Initialization CSC Specifications

2.2.1 Gateway Initialization CSC Groundrules

- SCID and TCID tables will be resident on the local hard drive.
- There will be only one Gateway SCID which will include the VxWorks kernels for each Gateway processor.
- The Gateway Initialization CSC will support the following Initialization Modes in order to synchronize the Gateway's boot sequence:

- Primitive
- SCID Initialization
- SCID/TCID Load
- Ready
- Operational

- The Gateway Initialization CSC will support the following initialization commands:
  - Init SCID
  - Init TCID
  - Activate Gateway
  - Terminate Gateway
  - Configuration Status

2.2.2 Gateway Initialization CSC Functional Requirements

The Functional Requirements for the Gateway Initialization CSC are arranged in the following major functions:

1. Primitive mode
2. SCID Initialization mode
3. SCID/TCID Initialization mode
4. Ready mode
5. Operational mode
6. Time board Initialization

2.2.2.1 Primitive Mode

The Primitive mode offers the most basic VxWorks operating system services. The Gateway will transition to the Primitive mode only if there is a VxWorks error during boot, or a Common Gateway Services software error during SCID Initialization. Primitive mode includes access to the Gateway via the Restricted Operational Network.

1. In case of a boot failure, or failure during SCID Initialization, the Gateway will transition to the Primitive Mode in order to allow a new SCID to be loaded.
2. The Primitive mode shall support communications (and FTP) via the Restricted Operational Network.
3. The Primitive mode shall support access to the Gateway's Hard Drive.
4. The Primitive mode shall not support communications via the RTCN.

2.2.2.2 SCID Initialization Mode

The SCID Initialization mode is a transitional mode between Gateway boot and the SCID/TCID Load mode. It is in the SCID Initialization mode that the Gateway loads the kernels from the Gateway Hard Drive and spawns the Initialization tasks for both the Gateway Control Processor (GCP), and the Front End Processor Controller (FEPC).

1. Gateway Initialization CSC shall record initialization messages on local storage media.
2. No external commands shall be accepted during the SCID Initialization mode.
3. During SCID Initialization, the Gateway Initialization CSC shall check the Power-Fail Flag in Non-volatile RAM before proceeding with Initialization (Post Thor).
4. During SCID Initialization, all Common Gateway Services CSCs shall be started.
5. During SCID Initialization, the Gateway Initialization CSC shall cyclically attempt to connect with the Ops CM server until a response is received.
6. Gateway Initialization shall receive Mode changes from the Gateway Processors (FEPCs) via the State Change Shared Memory Queue.

7. Gateway Initialization shall transition to SCID/TCID Load mode when SCID software load is complete, connection is established with the Ops CM Server, and the Gateway resident processors have changed to SCID/TCID Load mode.

2.2.2.3 SCID/TCID Load Mode

The SCID/TCID Load mode is the first point at which Ops CM commands are accepted and processed.

1. When in SCID/TCID Load mode, only Init SCID, Init TCID, and Configuration Status commands shall be accepted.
2. When a Configuration Status command is received, the Gateway Initialization CSC shall respond with the current Gateway Initialization mode, the current SCID Version, and the current TCID version (TBD).
3. Gateway Initialization CSC shall perform a reboot and enter SCID Initialization mode when the Init SCID command is received.
4. Gateway Initialization CSC shall perform all Table Loads and other TCID-specific actions when the Init TCID command is received.
5. Gateway Initialization CSC shall transition to Ready Mode when the Init TCID command is successfully completed, and the Gateway resident processors have changed to Ready mode.

2.2.2.4 Ready Mode

The Ready mode indicates that a TCID has been successfully loaded onto the Gateway. The next step is to Activate the Gateway to make it operational.

1. When in Ready mode, only Init SCID, Init TCID, Activate, and Configuration Status commands shall be accepted.
2. When in Ready mode, the Init SCID, Init TCID and Configuration Status commands shall be processed exactly as processed in the SCID/TCID Load mode.
3. Gateway Initialization CSC shall start Change Data generation tasks, System Message tasks, and other Gateway-specific tasks when the Activate command is received.
4. Gateway Initialization CSC shall transition to Operational mode when the Activate command is successfully completed, and the Gateway resident processors have changed to Operational mode.

2.2.2.5 Operational Mode

The Operational mode is the state in which the Gateway processes Commands, generates Responses, and outputs Change Data.

1. When in Operational mode, only Terminate, and Configuration Status Initialization commands shall be accepted.
2. When in Operational mode, the Configuration Status command shall be processed exactly as processed in the SCID/TCID Load mode.
3. Gateway Initialization CSC shall halt Change Data generation tasks, halt System Message tasks, halt other Gateway-specific tasks, and perform resource/memory deallocation when the Terminate command is received.
4. FTP will be disabled while in Operational mode.
5. Gateway Initialization CSC shall record all termination messages on local storage media.
6. Gateway Initialization CSC shall generate a system message prior to a successful termination (Terminate Gateway command).
7. Gateway Initialization CSC shall generate a system message if there is an error in termination.
8. Gateway Initialization CSC shall transition to Ready mode when the Terminate command is successfully completed, and the Gateway resident processors have changed to Ready mode.
9. If a Terminate command is received while the Gateway has Data Acquisition activated, it shall be treated as a Recovery situation. The Gateway shall perform a Recovery and transition to the Primitive mode.

2.2.2.6 Time board Initialization

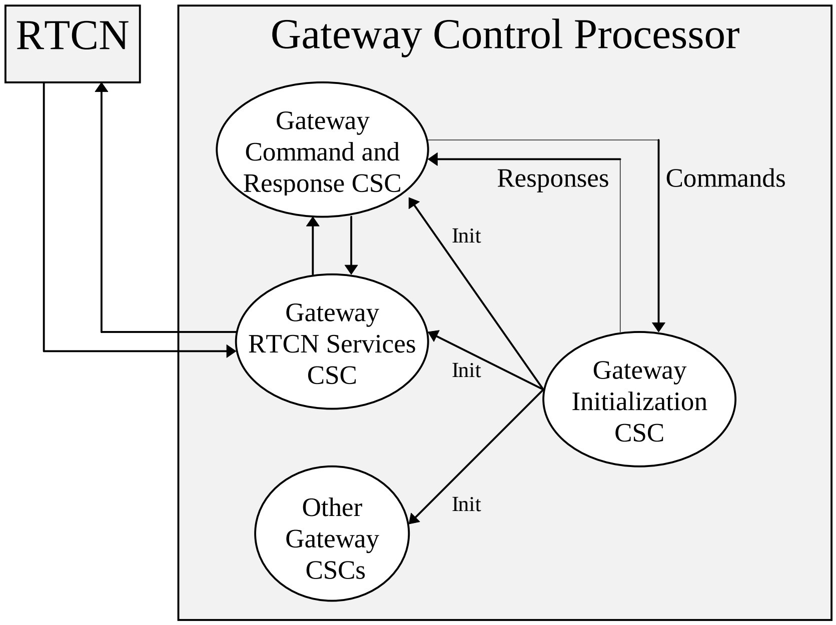
Resident in the Gateway is a time board which is referenced throughout processing for accurate time. The Time board must be configured during Initialization to insure that time is accurate when the Gateway is operational.

1. Gateway Initialization CSC shall be responsible for initializing the Time board.
2. If available, the Time board shall be configured to use the external IRIG-B signal in order to synchronize the time of day.
3. Gateway Initialization CSC shall program the Time board to interrupt the Gateway Control Processor at the System Synchronous Rate.

2.2.3 Gateway Initialization CSC Performance Requirements

No performance requirements have been identified for the Gateway Initialization CSC for the Thor delivery.

2.2.4 Gateway Initialization CSC Interfaces Data Flow Diagram



Gateway Initialization CSC is the first CSC to be spawned by the Gateway. It is responsible for the creation and termination of all other Gateway CSCs.

Gateway Initialization CSC commands and responses are handled through the Gateway Command and Response CSC.

2.3 Gateway Initialization CSC Design Specification

The Gateway Initialization CSC is responsible for the Initialization of all the Common Gateway functions. It has five Initialization modes through which it steps to achieve an Operational state for the Gateway. The five Initialization modes are Primitive mode, SCID Initialization mode, SCID/TCID Load mode, Ready mode, and Operational mode.

The Primitive mode is a means of catastrophic boot recovery in the Gateway. It consists of a minimal set of the VxWorks operating system software including the capability to FTP over the Restricted Operational Network (RON). A transition to the Primitive mode occurs when there is either an VxWorks operating system error during the VxWorks boot, or any error during the SCID Initialization. There are two options for the Primitive mode: Maintenance re-load of the SCID, or LDB Safing Sequences (to be implemented Post-Thor). The Maintenance re-load of the SCID must be done from the Console Port.

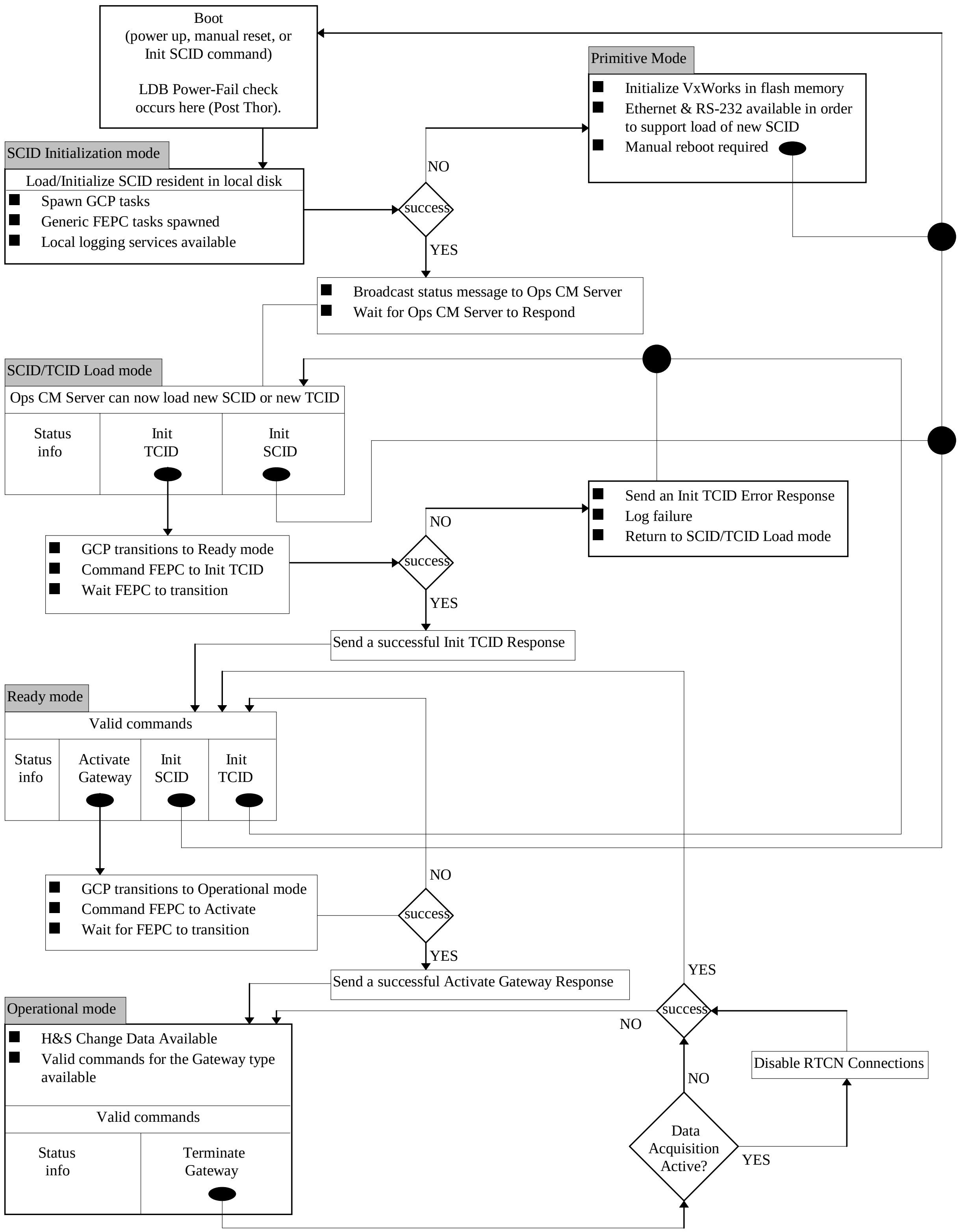
The SCID Initialization mode is a transitional mode in which all Common Gateway Services CSCs are spawned, and the initial RTCN connection is established. In this mode, all Subsystem Integrity Shared Memory Areas and Queues are set up. Log message, Error message, System message, System Event, Command and Response, and Maintenance User Interface services are all started in the SCID Initialization mode. After all CSCs have been started, the Gateway will transmit its boot status to the Ops CM Server over the RTCN. If the status is successful, the Ops CM will respond to the transmission, thereby establishing its connection with the Gateway. If an error occurs in the SCID Initialization mode, the Gateway will transition to the Primitive mode. Only when a successful connection with the Ops CM Server has been established will the Gateway complete the transition to SCID/TCID Load mode.

SCID/TCID Load mode is the first Initialization mode that can receive RTCN commands. Because communications with the Ops CM Server were established in the SCID Initialization mode, commands may now be accepted from that Server. In the SCID/TCID Load mode, the Configuration Status, SCID Init, and TCID Init commands are the only valid commands. The Configuration Status command returns to the sender the current Initialization status of the Gateway. Initialization mode, current SCID version, and current TCID name are all returned by this command. The SCID Init command initiates a reboot of the Gateway. The TCID Init command causes the load of TCID tables. A parameter in the TCID Init command is the TCID name. When the Gateway receives this command, it searches the Gateway Disk for that TCID's directory under /sd0:/tcid. If found, the tables under that directory are loaded onto the Gateway and a Successful response is returned to the sender. If there is an error loading the TCID tables, an unsuccessful response is generated, as is a System Message (if applicable). A successful TCID Init is the only way to transition to the Ready mode.

The Ready mode is reached when the TCID tables have been successfully loaded. Valid RTCN commands that may be received in this mode are Configuration Status, SCID Init, TCID Init, and Activate. The Configuration Status, SCID Init, and TCID Init commands operate exactly as defined above. The Activate command is the means to transition to the Operational mode. When an Activate command is received, the Gateway builds and opens all communication paths on the RTCN. Also, the transmission of Subsystem Integrity Change Data in the Change Data stream is started with the Activate command. If any paths are unsuccessful at opening, the Gateway will generate an Unsuccessful response to the sender and remain in the Ready mode. If all communication paths are successfully opened, the Gateway will return a Successful response and transition to the Operational mode.

The Operational mode is the state at which the Gateway is ready to begin processing. Only the Configuration Status and Terminate Ops CM commands are valid at this point. The Configuration Status command yields the same result described above. The Terminate command closes all Communication Paths, disables the Subsystem Integrity Change Data generation, and transitions the Gateway to the Ready mode. If the Terminate command is unsuccessful, the Gateway remains in the Operational mode. If a Terminate command is received while Data Acquisition is activated, all RTCN Interfaces are disconnected before returning to the Ready mode.

2.3.1 Gateway Initialization CSC Detailed Data Flow



2.3.2 Gateway Initialization CSC External Interfaces

2.3.2.1 Gateway Initialization CSC Message Formats

2.3.2.1.1 Init TCID Successful

Message Number =
Message Group = CGS
Severity = Informational

Gateway %s Init TCID successful.
TCID %s was loaded.
Gateway Name is %s.

Insert #1 = Text string	Gateway Host Name (e.g. ide1gwgse1)
Insert #2 = Text string	TCID Name
Insert #3 = Text string	Gateway Logical Name (e.g., GS1A)

2.3.2.1.2 Init TCID Error: Accessing Gateway Disk

Message Number =
Message Group = CGS
Severity = Error

Gateway %s Init TCID failed.
Gateway Disk was inaccessible.

Insert #1 = Text string	Gateway Host Name (e.g. ide1gwgse1)

2.3.2.1.3 Init TCID Error: Opening Data File

Message Number =
Message Group = CGS
Severity = Error

Gateway %s Init TCID failed.
File %s could not be opened.

Insert #1 = Text string	Gateway Host Name (e.g. ide1gwgse1)
Insert #2 = Text string	File Name on the Gateway Disk (e.g. act_data)

2.3.2.1.4 Activation successful

Message Number = 72
Message Group = CGS
Severity = Informational

%s Gateway Activation successful

Insert #1 = Text string	Gateway Logical Name (e.g. GS1A)

2.3.2.1.5 Activation failed

Message Number = 74
Message Group = CGS
Severity = Error

%s Gateway Activation failed

Insert #1 = Text string Gateway Logical Name (e.g. GS1A)

2.3.2.1.6 Termination successful

Message Number = 73
Message Group = CGS
Severity = Informational

%s Gateway Termination successful

Insert #1 = Text string Gateway Logical Name (e.g. GS1A)

2.3.2.1.7 Termination failed

Message Number = 75
Message Group = CGS
Severity = Error

%s Gateway Termination failed

Insert #1 = Text string Gateway Logical Name (e.g. GS1A)

2.3.2.2 Gateway Initialization CSC Display Formats

None.

2.3.2.3 Gateway Initialization CSC Input Formats

None.

2.3.2.4 Gateway Initialization CSC Recorded Data

None

2.3.2.5 Gateway Initialization CSC Printer Formats

None.

2.3.2.6 Gateway Initialization CSC Interprocess Communications

2.3.2.6.1 Gateway Configuration Status

CM Server Configuration Status Request (S CS <CPU>) (Routing Code 5, Request ID 1)

Bytes	C-C TO DESTINATION(S)	Bytes	RESPONSE FROM DESTINATION
	Header		Header
		1	Initialization Mode 0 = SCID Initialization (boot) 1 = SCID/TCID Load 2 = Ready 3 = Operational
		30	Current SCID Version (NULL terminated ASCII string)
		30	Current TCID Name (NULL terminated ASCII string)
		14	Name of TCID that is loaded (or 0)
		2	Number of TCIDs available that follow
		14	Name of TCID # 1
		:	:
		14	Name of TCID # n

Response Completion Codes:
0 Successful
800 Fail

2.3.2.6.2 Gateway to CM Server SCID Status Command

Gateway to CM Server SCID Init Status (Routing Code 5, Request ID 2)

Bytes	C-C TO DESTINATION(S)	Bytes	RESPONSE FROM DESTINATION
	Header		Header
1	= 0 = Successful = 1 = Unsuccessful (POST failed)		
30	Current SCID Version (NULL terminated ASCII string)		
30	Current TCID Version (NULL terminated ASCII string)		
14	Name of TCID that is loaded (or 0)		
2	Number of TCIDs available that follow		
14	Name of TCID # 1		
:	:		
14	Name of TCID # n		

Response Completion Codes:
0 Successful
800 Fail

2.3.2.6.3 CM Server Command to Init SCID or Init TCID

Initialize SCID or TCID (I SC <NODE>) (I TC <NODE>) (Routing Code 5, Request ID 3)

Bytes	C-C TO DESTINATION(S)	Bytes	RESPONSE FROM DESTINATION
	Header		Header
1	Lport to Init	30	Current SCID Version (NULL terminated ASCII string)

1	1 = Init SCID 2 = Init TCID	30	Current TCID Name (NULL terminated ASCII string)
30	SCID or TCID name (NULL terminated ASCII string)	30	Name of TCID that is loaded (or 0)
		2	Number of TCIDs available that follow
		30	Name of TCID # 1
		:	:
		30	Name of TCID # n

Response Completion Codes:
0 Successful
800 Fail

2.3.2.6.4 CM Server Command to Activate Gateway

CM Server Activate Platform Command (ACT <NODE>) (Routing Code 5, Request ID 4)

Bytes	C-C TO DESTINATION(S)	Bytes	RESPONSE FROM DESTINATION
	Header		Header

Response Completion Codes:
0 Successful
800 Fail

2.3.2.6.5 CM Server Command to Terminate Gateway

Terminate Subsystem (Routing Code 18, Request ID 1)

Bytes	C-C TO DESTINATION(S)	Bytes	RESPONSE FROM DESTINATION
	Header		Header

Response Completion Codes:
0 Successful
800 Fail

2.3.2.7 Gateway Initialization CSC External Interface Calls

None

2.3.2.8 Gateway Initialization CSC Table Formats

2.3.2.8.1 System Configuration Table (SCT)

The Gateway Initialization CSC is responsible for the initial load of the SCT into memory. The format of the SCT is TBD.

2.3.2.8.2 Activity Definition File

The activity file created and loaded by OPS CM as part of the gateway's TCID will have the following file name: /sd0:/tcid/act_data. The contents of the file will have the following format.

Line #	Field Description	Format	Max. Length
1	Activity Name	Null-terminated ASCII string	30
2	SCID Version	Null-terminated ASCII string	30
3	TCID Name	Null-terminated ASCII string	30
4	Tail ID	Null-terminated ASCII string	8
5	Flight Number	Null-terminated ASCII string	10
6	End Item Location	Null-terminated ASCII string	8

2.3.3 Gateway Initialization CSC Test Plan

2.3.3.1 Environment

The Common Gateway Services CIT will be performed in conjunction with the GSE Gateway Services CIT. A development Gateway will be configured as a GSE Gateway. RTCN commands will be sent to the development Gateway. The action taken and the responses returned will be verified.

2.3.3.2 Test Tools

The Gateway Integrated Test Environment CSC will be used to test the Gateway Initialization CSC.

2.3.3.3 Test Cases

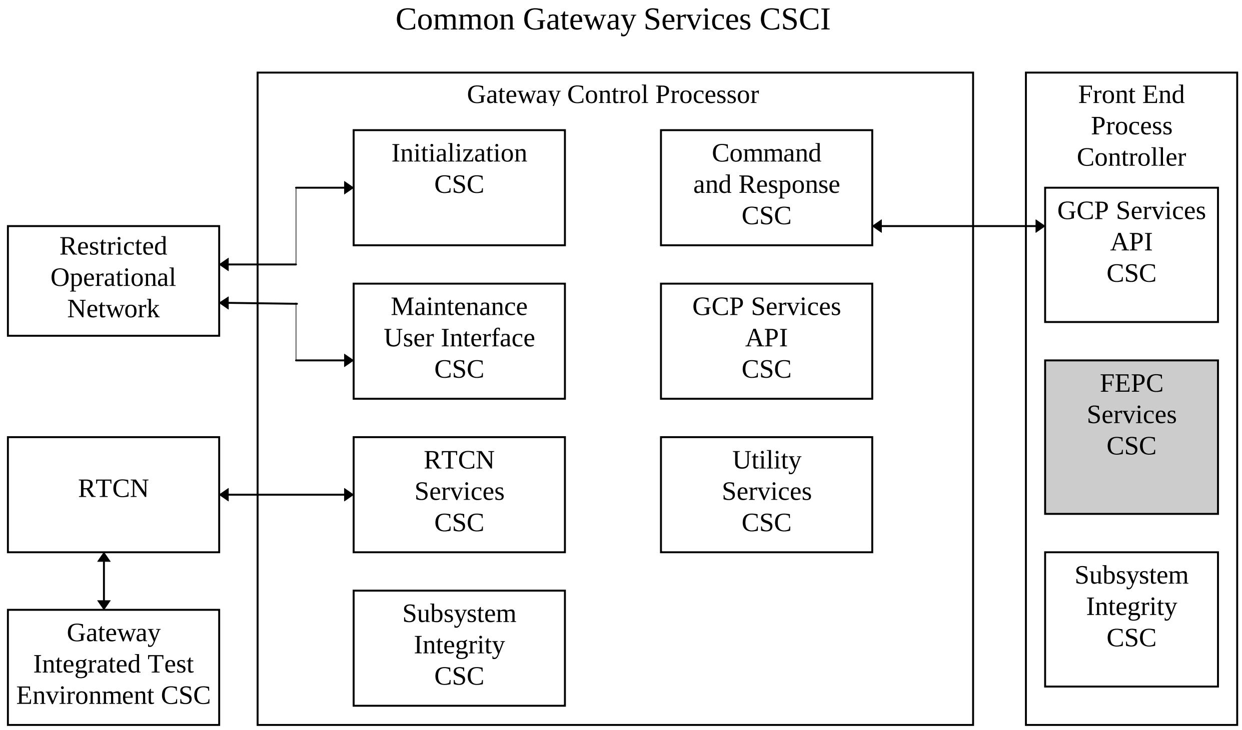
1. Verifying Primitive mode capabilities.
2. Sending an Init SCID Command in all Gateway Initialization modes.
3. Sending an Init TCID Command in all Gateway Initialization modes.
4. Sending an Activate Gateway Command in all Gateway Initialization modes.
5. Sending an Terminate Command in all Gateway Initialization modes.

3. Gateway FEPC Services CSC

3.1 Gateway FEPC Services CSC Introduction

3.1.1 Gateway FEPC Services CSC Overview

The Gateway FEPC Services CSC is the common software services provided to the Front End Process Controller (FEPC). The common initialization functions handle all sequencing involved to transition from power-on through operational mode. Gateway specific (GSE, LDB, PCM, etc.) routines will be called as part of the table load command and activate command.



3.1.2 Gateway FEPC Services CSC Operational Description

The Gateway FEPC Services CSC is initialized at subsystem startup by the vxWorks operating system. This CSC will start an initialization command processor task. The Gateway FEPC Services CSC will then wait for an initialization command. The TCID Init command will cause a gateway specific table load command to be executed. Successful completion of this function will cause a transition to ready mode (See mode description below). The Activate command will cause this CSC to terminate the tasks associated with common initialization and then call a gateway specific activation function. Successful completion of this function will cause a transition to operational mode (See mode description below). This CSC controls the initialization mode of the FEPC during the startup and load sequence. This mode information is used by all active CSC's to determine the validity of commands.

The measurement processing aspect of the Gateway FEPC Services CSC provides software that will take an input stream of unprocessed data that has been linked to a Measurement Descriptor Table entry and convert the raw data to processed engineering unit values. The data will then be sent to the RTCN using a Gateway GCP Services API call. This CSC will also provide a set of functions to the Command CSC for all rid and request id's supported by this CSC. The functions will perform the request and send a response via an API call. This CSC also contains Common

Table Load and Initialization Functions that load the Measurement Descriptor Tables into local memory. These functions are callable by each individual Gateway specific Table Load function.

3.2 Gateway FEPC Services CSC Specifications

Gateway FEPC Services is organized into the following major functions:
1. Common Initialization
2. Common Command Processor
3. Common Measurement Processor

3.2.1 Gateway FEPC Services CSC Groundrules

- The gateway specific CSCI (LDB, GSE, PCM, etc.) will provide the table load and activate functions.
- TCID tables must be loaded to the Gateway local media storage prior to initialization.

3.2.2 Gateway FEPC Services CSC Functional Requirements

3.2.2.1 Common Initialization

The Common Initialization aspect of Gateway FEPC Services CSC provides the Gateway specific software a means to track the Initialization sequence of the Gateway. This sequence is common among all Gateway types. Only the Table Load and Activate functions are unique across Gateway types. The unique functions are called by this Common Initialization.

1. Gateway FEPC Services will support the following operating modes:
   - 1.1. Initialization
   - 1.2. Load
   - 1.3. Ready
   - 1.4. Operational
2. Gateway FEPC Services will support the following load and initialize commands:
   - 2.1. Initialize SCID (reboot)
   - 2.2. Initialize TCID (load tables)
   - 2.3. Activate Gateway
   - 2.4. Terminate Gateway
3. Gateway FEPC Services will record initialization messages on local storage media.
4. Only initialization commands will be accepted during the Initialization mode.
5. Gateway FEPC Services will transition to Load mode when SCID software load is complete.
6. When in Load mode, only the Initialize SCID and Initialize TCID commands will be accepted.
7. Gateway FEPC Services will perform a reboot and enter Initialization mode when the Initialize SCID command is received.
8. Gateway FEPC Services will call a gateway type specific table load function as part of the Initialize TCID command.
9. Gateway FEPC Services will transition to Ready mode when the TCID tables are successfully loaded.
10. When in Ready mode, only the Activate Gateway and Terminate Gateway commands will be accepted.
11. Gateway FEPC Services will complete initialization and enter the Operational mode when the Activate Gateway command is received.
12. Gateway FEPC Services will call a gateway type specific activation function as part of the Activate Gateway command.
13. Gateway FEPC Services will terminate upon completion of a successful activation.
14. Gateway FEPC Services will provide a termination function to be called upon completion of the gateway type unique termination function.
15. The Gateway FEPC Services terminate function will restart the Gateway FEPC Services CSC and enter Ready mode.

16. Gateway FEPC Services will generate a system message prior to termination, whether due to an error or by command.
17. Gateway FEPC Services will record all termination messages on local storage media.
18. Gateway FEPC Services will be capable of generating system messages through the Gateway Common Services CSCI using the GCP Services API.
19. Gateway FEPC Services will be capable of requesting through the GCP Services API that a message be written to a file on the local hard drive or the local console port.
20. Gateway FEPC Services will provide the capability to be re-initialized without receiving a new table load command.

3.2.2.2 Common Command Processor

1. The Common Command Processor shall provide a task that will handle all incoming RTCN commands.
2. The Common Command Processor shall provide a function that allows all FEPC resources to correlate a function pointer to a Route Code and Request ID.
3. If an incoming command contains a Route Code and Request ID that has been registered, the Common Command Processor will call the function associated with that registration.

3.2.2.3 Common Measurement Processor

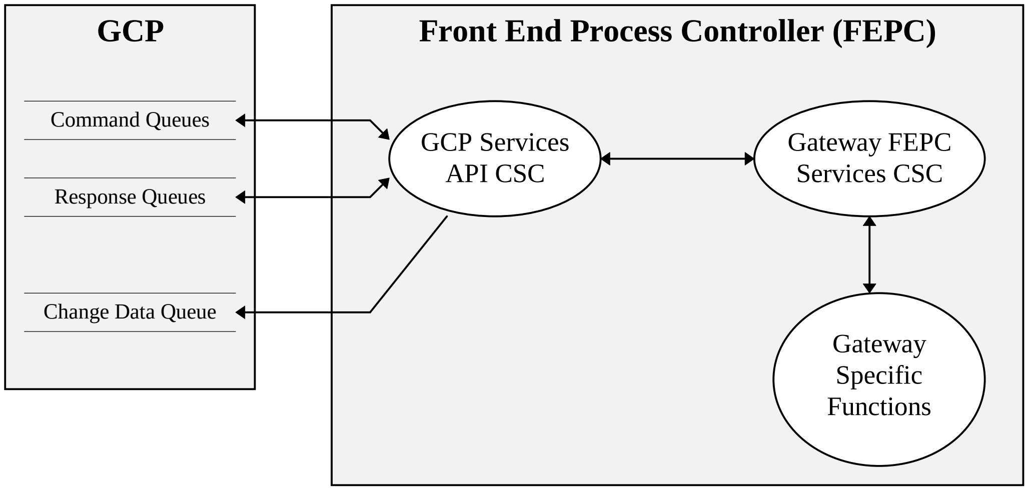
1. The Gateway shall convert measurements to standard IEEE-754 floating-point engineering unit form, if processing parameters for that measurement specify this conversion.
2. The Gateway shall provide the capability to convert to engineering units as specified by the conversion coefficients for each analog raw value.
3. The Gateway shall provide the capability to convert 32-bit single precision and 64-bit double precision GPC Floating point data.
4. The Gateway shall provide the capability to process digital pattern measurements composed of 2 to 64 bits.
5. The Gateway shall support the processing of parent words which contain multiple measurements by separating the parent word into measurements and processing each measurement individually.
6. The Gateway shall be capable of processing analog, digital pattern, and discrete group data types included within a parent word.
7. The Gateway shall process analog measurements based on type and/or subtype.
8. The Gateway shall provide the capability to process some or all bits of a discrete group.
9. *The Gateway shall provide the capability to process some or all of a discrete group from 1 to 64 bits.(Atlas)*
10. The Gateway shall process a group of individual data words defined as a Time Homogeneous Data Set (THDS).
11. The Gateway shall process words which make up a THDS that are not contiguous within a PCM frame.
12. The Gateway shall process all data types defined in the SLS (82K00200 Appendix A).
13. The Gateway shall maintain the current value of each measurement.
14. The Gateway shall compare each acquired measurement with the previous value of the measurement to detect a change in the measurement.
15. The Gateway shall perform change checks against unprocessed data.
16. The Gateway shall detect a change for each unprocessed measurement whenever the current sample and the previously recorded sample is different.
17. The Gateway shall provide the capability to test each discrete measurement to detect any change in data.
18. The Gateway shall send out the current value of a measurement when the first sample is acquired.
19. Gateway FEPC Services CSC shall load all required TCID tables from the local hard drive when the Initialize TCID command is received.
20. Gateway FEPC Services CSC shall respond to the Initialize TCID command with a success or fail status.

21. Gateway FEPC Services CSC shall be capable of generating system messages through the Gateway Common Services CSCI using the GCP Services API.
22. Gateway FEPC Services CSC shall be capable of requesting through the GCP Services API that a system message be written to a file on the local hard drive, the local console port, or the RTCN interface.
23. Gateway FEPC Services CSC shall perform the following verification checks on the loaded tables:
    23.1. Correct entries per record will be checked.
    23.2. EU coefficients which are used by the CMDT shall be tested to ensure at least a first order polynomial is present.

3.2.3 Gateway FEPC Services CSC Performance Requirements

1. The Gateway FEPC Services CSC's Common Measurement Processor shall process 30,000 measurements per second.

3.2.4 Gateway FEPC Services CSC Interfaces Data Flow Diagram



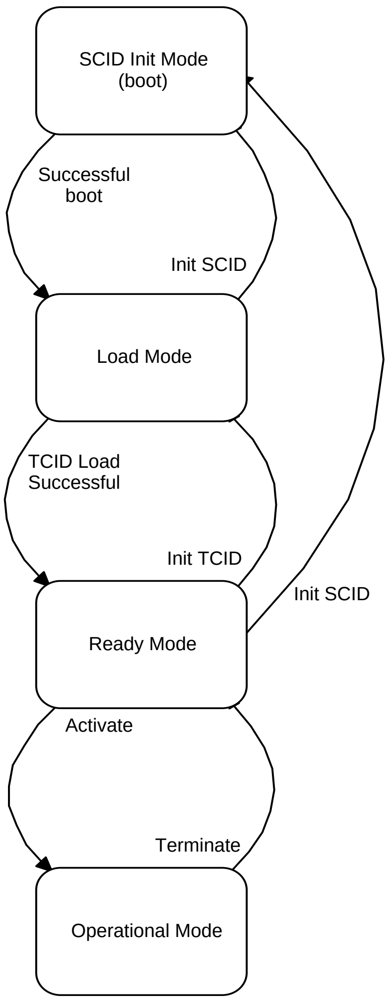
The Gateway FEPC Services CSC accepts commands from the RTCN via the GCP Services API CSC which is part of Common Gateway Services CSCI..

The Gateway FEPC Services CSC calls a gateway type unique table load function as part of the Initialize TCID command.

The Gateway FEPC Services CSC calls a gateway type unique activate function as part of the Activate command. The Measurement Processing CSC interfaces with a number of processes. The main interface is the incoming raw data stream that contains unprocessed data and a pointer to the MDT from the Gateway Interface CSC.The Measurment Processing CSC is spawned by the Initialization CSC. Commands to control the processing of measurements will come from the Table Maintenance Function. The Table Maintenance Function will also co-ordinate with the Measurement Processing CSC when a change to the tables is made. The Measurement Descriptor Tables are used to determine what the incoming raw data is and how to process it.

3.3 Gateway FEPC Services CSC Design Specification

3.3.1 Common Initialization Modes



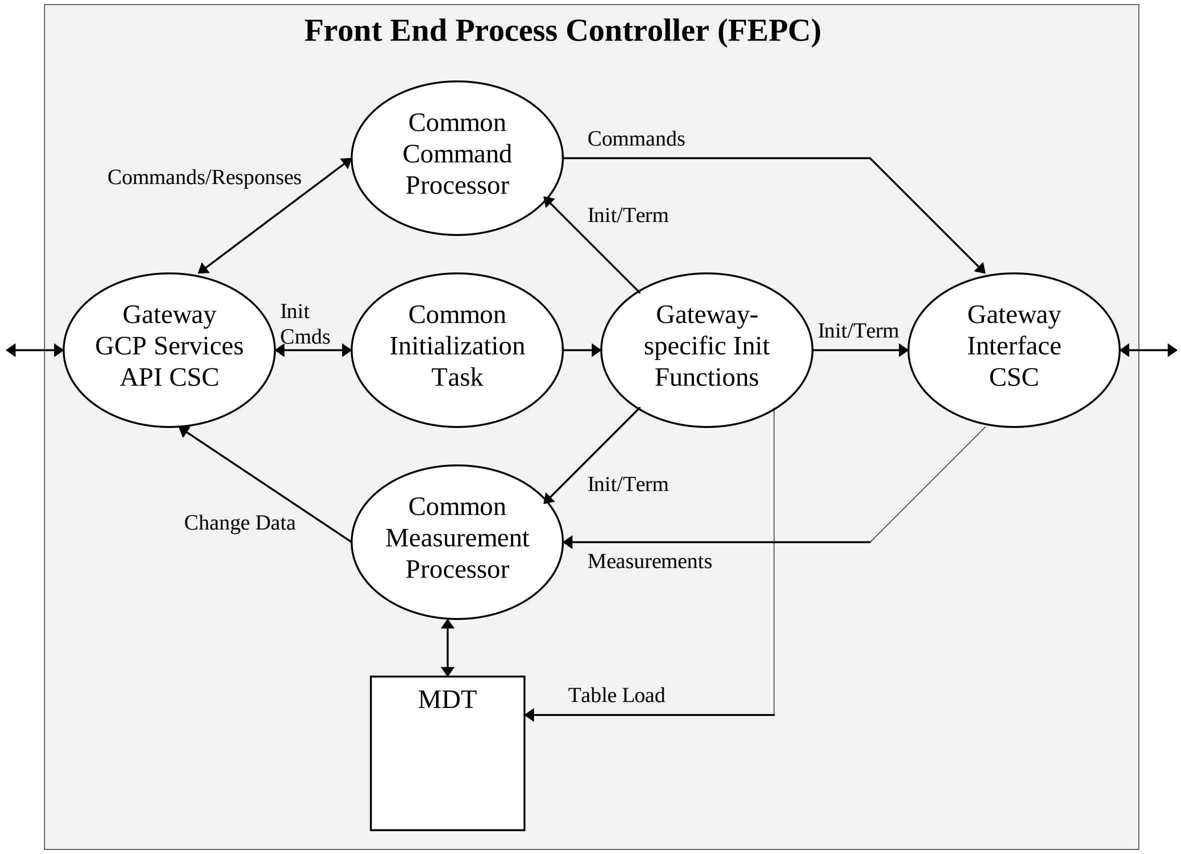
SCID Initialization Mode
This CSC is made up of a single task which is spawned by the vxWorks operating system at startup/boot time (SCID Init Mode is entered as part of the boot process). Upon successful completion of the boot process, the FEPC will transition to Load Mode (Ref. Figure 1.3).

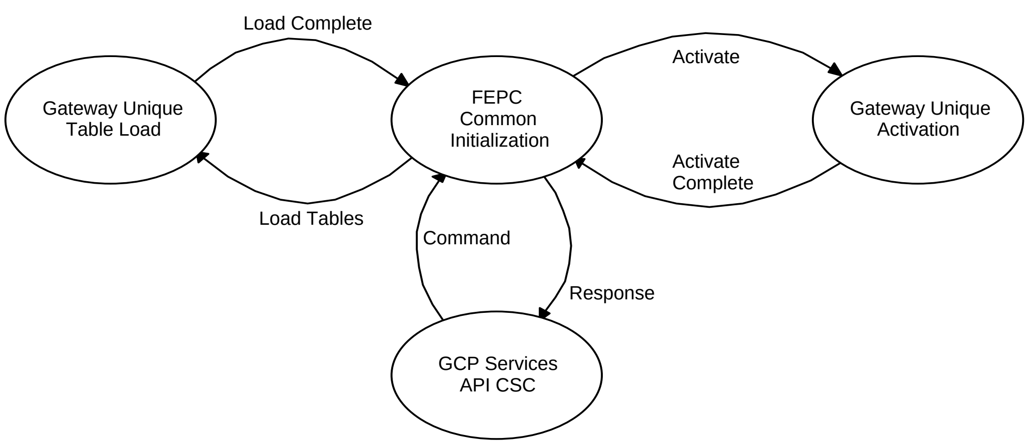
Load Mode
This CSC will accept the Initialize SCID and Initialize TCID commands while in Load Mode. The purpose of this mode is to allow the OPS/CM Server/Master Console to update the SDID and TCID files on the hard drive if required. The Init SCID command will cause the FEPC to reboot. This command will be required if new software is loaded. The Init TCID command is forwarded to a gateway specific table load function which will load the TCID tables from the local hard drive. If this process is successful the table load function will call the Load Complete function provided by this CSC which will cause a transition to Ready Mode. The gateway specific table load function is responsible for generating the response to the Load TCID command

Ready Mode
This CSC will accept the Activate, Initialize SCID and Initialize TCID commands while in this mode. The Init SCID command will cause the FEPC to reboot. The Init TCID command will cause a transition back to Load Mode and the tables to be reloaded. The Activate command will cause this CSC to call a gateway specific activation function. The gateway specific activation function will call the Activate Complete function of this CSC which will cause a transition to Operational Mode. The gateway specific activation function is responsible for generating the response to the Activate command

Operational Mode
The Gateway FEPC Services CSC will terminate upon completion of a successful activation. The gateway unique command processor then becomes responsible for accepting all RTCN commands. A terminate function is provided by this CSC which will be called by the gateway unique command processor upon completion of the gateway unique termination activity. This function will reactivate the Gateway FEPC Services CSC.

3.3.2 Gateway FEPC Services CSC Detailed Data Flow





The Gateway FEPC Services CSC is initialized at subsystem start up by the vxWorks operating system. This CSC will initiate a command processor task which will wait for an initialization command.

The Initialize TCID command will cause a gateway specific table load function to be executed. This function will take whatever action is required to load and verify the gateway unique tables. Upon completion of table load, the table load function must call the init_table_load_complete() function defined by this CSC. Successful completion of the table load will cause a transition to ready mode (See mode description below)

The Activate command will cause this CSC to call a gateway specific activation function . This function will take whatever action is required to transition the gateway to an operational mode that is unique to the gateway type. Upon completion of this transition, the activate function must call the init_activate_complete() function defined by this CSC. Successful completion of this function will terminate the tasks associated with common initialization and cause a transition to operational mode (See mode description below)

This CSC controls the initialization mode of the FEPC during the startup and load sequence. This mode information is used to determine the validity of commands.

3.3.3 Gateway FEPC Services CSC External Interfaces

3.3.3.1 Gateway FEPC Services CSC Message Formats

None - all system messages related to Gateway FEPC Services CSC will be generated by the Gateway Initialization CSC or the FEPC specific load, activate, and terminate function calls.

3.3.3.1.1 Common Measurement Processor

3.3.3.1.1.1 Change Data Queue Full

Message Number =
Message Group = CGS
Severity = Error

Change data lost - change data queue is full

A change data write could not be completed because the change data queue is full.

3.3.3.1.1.2 Queue Initialization Error

Message Number =
Message Group = CGS
Severity = Error

Unable to initialize queue for Measurement Processor.

3.3.3.1.1.3 Memory Allocation Error

Message Number =
Message Group = __________
Severity = Error

Unable to allocate memory for %s table.

Insert #1	Text String	Table Name • Format Descriptor Table • Telemetry Descriptor • Format List • Measurement Descriptor Table • Analog 32bit Table • Analog 64bit Table • Digital Pattern Table • Discrete Parent Table

3.3.3.1.1.4 File Open Failure

Message Number =
Message Group = __________
Severity = Error

Unable to open file %s

Insert #1	Text String	File Name • PCM files • Measurement Descriptor Files

3.3.3.1.1.5 File Scan Failure

Message Number =
Message Group = CGS
Severity = Error

Scan error reading file %s at line %d

Insert #1	Text String	File name • an32.mdt • an64.mdt • dp.mdt • dpat.mdt
Insert #2	Integer	Line number

Details:

An error occurred while reading the specified file at the indicated line. Either the number of elements in the record was incorrect or an unexpected end-of-file was encountered.

3.3.3.2 Gateway FEPC Services CSC Interprocess Communications

3.3.3.2.1 Common Initialization

3.3.3.2.1.1 Activate

Activate the gateway (transition to operational mode).

GSE Activate Data Acquisition (Routing Code 8, Request ID 1)

Bytes	C-C TO DESTINATION(S)	Bytes	RESPONSE FROM DESTINATION
	Header		Header
2	= 1 = Activate = 2 = Start Processing = 3 = Activate and Start Processing		
2	MSB = 1 = enable Type II polling = 0 = enable Type I polling		

Response Completion Codes:
0 Successful
1 Fail

3.3.3.2.1.2 Init/Load

Reboot (Load SCID) or load and verify TCID tables.

Load Gateway SCID or TCID (Routing Code 9, Request ID 1)

Bytes	C-C TO DESTINATION(S)	Bytes	RESPONSE FROM DESTINATION
	Header		Header
1	Spare		
1	= 1 = Gateway Load SCID = 2 = Gateway Load TCID		
14	SCID or TCID Name		

Response Completion Codes:
0 Successful
1 Fail

3.3.3.3 Gateway FEPC Services CSC Display Formats

None.

3.3.3.4 Gateway FEPC Services CSC Input Formats

None.

3.3.3.5 Gateway FEPC Services CSC Recorded Data

None

3.3.3.6 Gateway FEPC Services CSC Printer Formats

None.

3.3.3.7 Gateway FEPC Services CSC External Interface Calls

3.3.3.7.1 Common Initialization Interfaces

FEPC initialization is designed to transition the FEPC from power on through operational configuration and also through termination.

Services provided by FEPC common initialization include:
- Manages all initialization functions from power on boot through termination.
- Performs NFS mount of GCP’s hard drive as /sd0:
- Performs processing of the Init SCID command.
- Provides an interface to a FEPC unique table load function.
- Provides an interface to a FEPC unique activate function.
- Provides an interface to a FEPC unique terminate function.

Each unique FEPC type (GSE, LDB, PCM, etc.) must provide a table load function, an activate function and a terminate function. These functions will be called during processing of the Init TCID, Activate and Terminate commands respectively.

3.3.3.7.1.1 FEPC-unique table load function.

For consistency, the name of the function should be:

<gateway type>TableLoad (e.g. gseTableLoad).

The calling convention for the function is:

```
void xxxTableLoad (        CGS_COMMAND_INFO_TYPE *info,
                           void *body,
                           char *directory);
```

info and *body* are pointers to the header information structure and command body as defined in the GCP services API document.

directory is a pointer to a character string which contains the path to the TCID files to be loaded. The string is of the form "/sd0:/tcid/XXXXX/" where XXXXX is the TCID name passed by the Init TCID command. The table load task should copy this string and append the file name prior to opening the file for input.

The user provided table load function should spawn or communicate with a FEPC unique table load task return immediately. FEPC common initialization will wait for an cgsTableLoadComplete() call to be performed by the table load task before accepting any further initialization commands. Commands received during a table load will be rejected with a TABLE_LOAD_IN_PROGRESS return code.

The user provided table load task is responsible for:
1. Sending an ISSUED RESPONSE if required
2. Loading and verifying all TCID tables.
3. Sending the final command success or fail response.
4. Sending any system messages required due to errors.
5. Calling the cgsTableLoadComplete(int status) function with status being OK or ERROR.

When OK status is passed to cgsTableLoadComplete(),FEPC Common Initialization will transition to READY mode. ERROR status will cause the gateway to remain in LOAD mode

3.3.3.7.1.2 FEPC-unique activate function.

For consistency, the name of the function should be:
<gateway type>Activate (e.g. gseActivate).

The calling convention for the function is:

```
void xxxActivate (         CGS_COMMAND_INFO_TYPE *info,
                           void *body);
```

info and *body* are pointers to the header information structure and command body as defined in the GCP services API document.

The user provided activate function should spawn or communicate with an activation task and return immediately.

The user provided activation task is responsible for:
1. Sending an ISSUED RESPONSE if required.
2. Performing all task spawns, queue creation, etc. required to become a functioning gateway.
3. Sending the final success or fail command response.
4. Sending any system messages required due to errors.
5. Calling the cgsActivateComplete(int status) function with status being OK or ERROR.

When OK status is passed to cgsActivateComplete(), FEPC Common Initialization will transition to OPERATIONAL mode. ERROR status will cause the gateway to remain in READY mode.

3.3.3.7.1.3 FEPC-unique termination function

For consistency, the name of the function should be:

<gateway type>Terminate (e.g. gseTerminate).

The calling convention for the function is:

```
void xxxTerminate (      CGS_COMMAND_INFO_TYPE *info,
                         void *body);
```

info and *body* are pointers to the header information structure and command body as defined in the GCP services API document.

The user provided terminate function should spawn or communicate with an termination task and return immediately.

The user provided termination task is responsible for:
1. Cleaning up all FEPC unique resources (tasks, queues, semaphores, etc)
2. Calling the cgsTerminateComplete(int status) function
3. Terminating itself.

When OK status is passed to cgsTerminateComplete(),FEPC Common Initialization will:
1. Clear the command list of all route codes which are gateway unique. Only initialization commands will remain.
2. Transition to READY mode

3.3.3.7.2 Common Command Processor

Common Command Processor will provide the front end interface to RTCN commands. The command decode task will take commands from the GCP command queues, decode the route code and request id and call a user defined function which is specific to the route code / request id pair. An interface is provided to allow each FECP CSC to register the commands that it is responsible for.

3.3.3.7.2.1 Command Functions

All user provided functions called by the decode task must have the form:

```
void functionName(       CGS_COMMAND_INFO_TYPE *info,
                         void *body);
```

info and *body* are pointers to the header information structure and command body as defined in the API section later in this document.

The command function is responsible for:
1. Performing any validation required
2. Sending the issued response if required
3. Executing the command
4. Generating the command response
5. Sending any system message required due to errors

The user provided command function should return as soon as possible so the command task can accept the next command. The command function may completely execute the command if processing time is minimal. If processing time is significant, the command function should copy the info and body information and pass them to a task which executes the command. The info and body information must be copied since the command decode task will reuse the info and body buffers for the next command.

3.3.3.7.2.2 Registering Commands

The following function is provided to allow FECP CSC’s to register commands with the decode task.

```
STATUS cgsRegisterCommand (  UNSIGNED8 routeCode,
                             UNSIGNED8 requestId,
                             UNSIGNED16 criticality,
                             void (cmdFunction*) () );
```

Description	Registers a route code / request id pair with the command decode task and the GCP.		
Parameters	routeCode	Command route code	
	requestId	Command request ID	
	criticality	0 = Never critical	The command must never be critical. The GCP will reject the command if the critical flag is set in the command header.
		1 = Allow critical	The command is allowed to be critical. The GCP will place the command on the high or normal priority queue based on the setting of the criticality flag in the command header.
		2 = Always critical	The GCP will always place the command on the high priority queue, independent of the criticality flag in the command header
	cmdFunction	A pointer to the function to be executed when this route code / request id pair is received. The function must be of the form defined in the previous section	
Returns	OK or ERROR as defined in vxWorks.h		

3.3.3.7.2.3 Terminate the Common Command Processor

STATUS cgsTermCP();

Description	Terminate the FEPC command decoder so that a user task can perform the command decode function.
Parameters	None
Returns	OK or ERROR as defined in vxWorks.h

3.3.3.7.2.4 Restart the Common Command Processor

STATUS cgsInitCP();

Description	Restart the FEPC command decoder after a termination so that it can process the initialization commands.
Parameters	None
Returns	OK or ERROR as defined in vxWorks.h

3.3.3.7.3 Common Measurement Processor

3.3.3.7.3.1 Initialize Measurement Processing

STATUS cgsMeasProcInit();

Description: Measurement process initialization. This process sets up the input data queue for the measurement processor and initializes the process control variables

Parameters: None

Returns: OK or ERROR as defined in vxWorks.h

3.3.3.7.3.2 AM_AU Conversion Function

FLOAT32 AM_AU_Proc (UNSIGNED16 counts)

Description: Convert a 16-bit analog raw count value to an engineering unit (IEEE 754 single precision floating point) Engineering Unit value.

Parameters: counts

Analog Unipolar (AU)

- For KSC GSE the left most bit is the MSB
- For LDB and PCM the second bit is the MSB
- The magnitude bits describe a decimal integer in binary

Returns: Raw Value in 32 bit IEEE floating point

3.3.3.7.3.3 AM_AB Conversion Function

FLOAT32 AM_AB_Proc (UNSIGNED16 counts)

Description: Convert a 16-bit analog raw count value to an engineering unit (IEEE 754 single precision floating point) Engineering Unit value.

Parameters: counts

Analog Bipolar (AB)

- The left most bit is the sign bit
- The next bit is the MSB
- The right bit is the LSB
- The magnitude bits describe a decimal integer in binary
- If the sign bit is negative (1) then the magnitude is in two’s compliment form

Returns: Raw Value in 32 bit IEEE floating point

3.3.3.7.3.4 AM_AHU Conversion Function

FLOAT32 AM_AHU_Proc (UNSIGNED16 counts)

Description: Convert a 16-bit analog raw count value to an engineering unit (IEEE 754 single precision floating point) Engineering Unit value.

Parameters: counts Halfword Unsigned (AHU)

- The left most bit is the MSB
- The magnitude bits describe a decimal number in binary

Returns: Raw Value in 32 bit IEEE floating point

3.3.3.7.3.5 AM_ASM Conversion Function

FLOAT32 AM_ASM_Proc (UNSIGNED16 counts)

Description: Convert a 16-bit analog raw count value to an engineering unit (IEEE 754 single precision floating point) Engineering Unit value.

Parameters: counts Bit String Magnitude (ASM)

- The left most bit is the sign bit
- The second bit is the MSB
- The magnitude bits describe a decimal number in binary

Returns: Raw Value in 32 bit IEEE floating point

3.3.3.7.3.6 AM_AOS Conversion Function

FLOAT32 AM_AOS_Proc (UNSIGNED16 counts)

Description: Convert a 16-bit analog raw count value to an engineering unit (IEEE 754 single precision floating point) Engineering Unit value.

Parameters: counts Halfword Overflow Signed (AOS)

- The left most bit is the sign bit
- The second bit is the overflow bit
  1 = overflow 0 = No overflow
- The third bit is the MSB

The magnitude bits describe a decimal number in binary

Returns: Raw Value in 32 bit IEEE floating point

3.3.3.7.3.7 AM_BCD Conversion Function

FLOAT32 AM_BCD_Proc (UNSIGNED16 counts)

Description: Convert a 16-bit analog raw count value to an engineering unit (IEEE 754 single precision floating point) Engineering Unit value.

Parameters: counts Binary Coded Decimal (BCD)

- The start bit is the MSB
- The magnitude bits are a four bit BCD code describing a decimal number starting with the LSB

Returns: Raw Value in 32 bit IEEE floating point

3.3.3.7.3.8 AM_TAB Conversion Function

FLOAT32 AM_TAB_Proc (UNSIGNED16 counts)

Description: Convert a 16-bit analog raw count value to an engineering unit (IEEE 754 single precision floating point) Engineering Unit value.

Parameters: counts TACAN Bearing Word (TAB)

- MMDB BSE data type
- 2-15 bits in length
- Has a data range and units
- Originates from TAC card
- Card and channel location is zero
- The start bit is the MSB

Returns: Raw Value in 32 bit IEEE floating point

3.3.3.7.3.9 AM_AMF Conversion Function

FLOAT32 AM_AMF_Proc (UNSIGNED16 counts)

Description: Convert a 1-bit analog raw count value to an engineering unit (IEEE 754 single precision floating point) Engineering Unit value.

Parameters: counts Analog Measurement Filter (AMF)

- Has a length of 1
- The Gateway will compute a 16 bit bipolar analog filtered value for ET OI point sensors
- Valid on all PCM links
- Has an associated AMF constant whose value is less than 1 and greater than zero; up to 6 digits to the right of the decimal point may be specified
- Significant change parameter is invalid for this subtype

Returns: Raw Value in 32 bit IEEE floating point

3.3.3.7.3.10 AMDP_AU Conversion Function

FLOAT64 AMDP_AU_Proc (UNSIGNED64 counts)

Description: Convert a 17-64 bit analog raw count value to an engineering unit (IEEE 754 double precision floating point) Engineering Unit value.

Parameters: counts Analog Unipolar (AU)

- For KSC GSE the left most bit is the MSB
- For LDB and PCM the second bit is the MSB
- The magnitude bits describe a decimal integer in binary

Returns: Raw Value in 32 bit IEEE floating point

3.3.3.7.3.11 AMDP_AB Conversion Function

FLOAT64 AMDP_AB_Proc (UNSIGNED64 counts)

Description: Convert a 17-64 bit analog raw count value to an engineering unit (IEEE 754 double precision floating point) Engineering Unit value.

Parameters: counts Analog Bipolar (AB)

- The left most bit is the sign bit
- The next bit is the MSB
- The right bit is the LSB
- The magnitude bits describe a decimal integer in binary
- If the sign bit is negative (1) then the magnitude is in two’s compliment form

Returns: Raw Value in 32 bit IEEE floating point

3.3.3.7.3.12 AMDP_TAC Conversion Function

FLOAT32 AMDP_TAC_Proc (UNSIGNED16 counts)

Description: Convert a 18-bit analog raw count value to an engineering unit (IEEE 754 double precision floating point) Engineering Unit value.

Parameters: counts TACAN Range Word (TAC)

- The length must be 18 bits
- The start bit is the MSB
- The magnitude bits are assumed to be a four bit code defining a decimal number starting with the LSB

Returns: Raw Value in 32 bit IEEE floating point

3.3.3.7.3.13 FP_SPL Conversion Function

FLOAT64 FP_SPL_Proc (UNSIGNED32 gpc_spl)

Description: Convert a GPC Floating Point value to an engineering unit (IEEE 754 double precision floating point) Engineering Unit value.

Parameters: gpc_fp

Single Precision (SPL)

- Length is 32 bits
  A floating point number consisting of an unsigned characteristic (or exponent) and a signed fraction; the quantity expressed by this number is the product of the fraction and the number 16 raised to the power of the exponent

  where:

  Bit 00 = Sign of Fraction
  0 = Positive
  1 = Negative

  Bit 01-07 = Characteristic - A binary number with a range of 0 through 127. The exponent is derived by subtracting 64 from the characteristic. Therefore, the range of the exponent is -64 through +63.

  Bit 08-31 = Fraction - A binary number less than 1 with the binary point to the left of the high order digit (between bit 7 and 8)

Returns: Raw Value in 32 bit IEEE floating point

3.3.3.7.3.14 FP_HPL Conversion Function

FLOAT32 FP_HPL_Proc (UNSIGNED16 gpc_hpl)

Description: Convert a GPC Floating Point value to an engineering unit (IEEE 754 single precision floating point) Engineering Unit value.

Parameters: gpc_hpl

Half Precision (HPL)

- Length is 16 bits
- In CCMS this subtype was treated by GOAL and the CCMS Command Processor as an SPL subtype

  where:

  Bit 00 = Sign of Fraction
  0 = Positive
  1 = Negative

  Bit 01-07 = Characteristic - A binary number with a range of 0 through 127. The exponent is derived by subtracting 64 from the characteristic. Therefore, the range of the exponent is -64 through +63.

Bit 08-15 = Fraction - A binary number less than 1 with the binary point to the left of the high order digit (between bit 7 and 8)

Returns: Raw Value in 32 bit IEEE floating point

3.3.3.7.3.15 FP_DPL Conversion Function

FLOAT32 FP_DPL_Proc (UNSIGNED64 gpc_dpl)

Description: Convert a GPC Floating Point value to an engineering unit (IEEE 754 double precision floating point) Engineering Unit value.

Parameters: gpc_dpl

Double Precision (DPL)

- Length is 64 bits

A floating point number consisting of an unsigned characteristic (or exponent) and a signed fraction; the quantity expressed by this number is the product of the fraction and the number 16 raised to the powerof the component

Bit 00 = Sign of Fraction
0 = Positive
1 = Negative

Bit 01-07 = Characteristic - A binary number with a range of 0 through 127. The exponent is derived by subtracting 64 from the characteristic. Therefore, the range of the exponent is -64 through +63.

Bit 08-63 = Fraction - A binary number less than 1 with the binary point to the left of the high order digit (between bits 7 and 8)

Returns: Raw Value in 64 bit IEEE floating point

3.3.3.7.3.16 FP_EPL Conversion Function

FLOAT32 FP_EPL_Proc (UNSIGNED64 gpc_epl)

Description: Convert a GPC Floating Point value to an engineering unit (IEEE 754 double precision floating point) Engineering Unit value.

Parameters: gpc_epl

Extended Precision (EPL)

- Length is 48 bits

A floating point number consisting of an unsigned characteristic (or exponent) and a signed fraction; the quantity expressed by this number is the product of the fraction and the number 16 raised to the power of the component

Bit 00 = Sign of Fraction
0 = Positive
1 = Negative

Bit 01-07 = Characteristic - A binary number with a range of 0 through 127. The exponent is derived by subtracting 64 from the characteristic. Therefore, the range of the exponent is -64 through +63.

Bit 08-47 = Fraction - A binary number less than 1 with the binary point to the left of the high order digit (between bits 7 and 8)

Returns: Raw Value in 64 bit IEEE floating point

3.3.3.7.3.17 Load MDT Tables Command

STATUS **cgsMdtTableLoadTask(char * path);**

Description:	This function is called from the pcmTableLoad function and loads all the Measurement Descriptor records from the disk files. Internal data structures are allocated, initialized and populated.	
Parameters:	path	A pointer to a character string containing the path to the directory where the TCID tables are located. The path string is terminated by a “/” character
Returns:	OK or ERROR as defined by vxWorks.h	

3.3.3.7.3.18 Unload MDT Tables Command

void **cgsMdtTableFreeTask();**

Description:	This function is called from the pcmTableFree function and will free all the memory used for the Measurement Descriptor Tables Gateway tables.	
Parameters:	None	
Returns:	Nothing	

3.3.3.7.3.19 Activate/Inhibit FD Processing

```
void aiFdProc(    GCPS_COMMAND_INFO_TYPE *info,
                  void *body);
```

Description: Activates or inhibits data processing of the specified FDID. The FDID will continue to be polled at the HIM

Parameters: info Pointer to a GCPS_COMMAND_INFO_TYPE structure as defined by the GCP Common Services API.

body Pointer to the message body.

Returns: None

3.3.3.7.3.20 Activate/Inhibit Global Processing

```
void aiProcAll(    GCPS_COMMAND_INFO_TYPE *info,
                   void *body);
```

Description:	Activates or inhibits data processing globally.	
Parameters:	info	Pointer to a GCPS_COMMAND_INFO_TYPE structure as defined by the GCP Common Services API.
	body	Pointer to the message body.
Returns:	None	

3.3.3.7.3.21 Read Engineering Unit Polynomial

```
void readEuPoly( GCPS_COMMAND_INFO_TYPE *info,
                 void *body);
```

Description:	Reads the Engineering Unit Polynomial coefficients for the specified FDID.	
Parameters:	info	Pointer to a GCPS_COMMAND_INFO_TYPE structure as defined by the GCP Common Services API.
	body	Pointer to the message body.
Returns:	None	

3.3.3.7.3.22 Activate/Inhibit FD Change Check

```
void aiChangeCheck(    GCPS_COMMAND_INFO_TYPE *info,
                       void *body);
```

Description:	Activates or inhibits change checking for the specified FDID. An FDID with change checking inhibited will be output every sample.	
Parameters:	info	Pointer to a GCPS_COMMAND_INFO_TYPE structure as defined by the GCP Common Services API.
	body	Pointer to the message body.
Returns:	None	

3.3.3.7.3.23 Activate/Inhibit Global Change Check

```
void aiChangeCheckAll(   GCPS_COMMAND_INFO_TYPE *info,
                         void *body);
```

Description:	Activates or inhibits change checking globally. When change checking is inhibited, all samples will be output.	
Parameters:	info	Pointer to a GCPS_COMMAND_INFO_TYPE structure as defined by the GCP Common Services API.
	body	Pointer to the message body.
Returns:	None	

3.3.3.7.3.24 Change Engineering Unit Polynomial

void changeEuPoly(GCPS_COMMAND_INFO_TYPE *info,
void *body);

Description: Changes the Engineering Unit Polynomial coefficients for the specified FDID..

Parameters: info Pointer to a GCPS_COMMAND_INFO_TYPE structure as defined by the GCP Common Services API.

body Pointer to the message body.

Returns: New value and old value of EU polynomial coefficients

3.3.3.7.3.25 Status Function Designator

void statusFd(GCPS_COMMAND_INFO_TYPE *info,
void *body);

Description: Read the mdt of a FDID

Parameters: info Pointer to a GCPS_COMMAND_INFO_TYPE structure as defined by the GCP Common Services API.

body Pointer to the message body.

Returns: Returns the measurement descriptor table information for a FDID

3.3.3.8 Gateway Initialization CSC Table Formats

3.3.3.8.1 Common Measurement Processor

3.3.3.8.1.1 Measurement Descriptor Table (MDT)

Data Type	*Item*	*Description*
PROC_FLAGS	proc_flags;	Processing flags
UNSIGNED32	fdid;	Function Designator ID
UNSIGNED32	*ptr;	Pointer to measurement specific table
UNSIGNED32	type;	Measurement Type
UNSIGNED32	sub_type;	Measurement Sub Type
UNSIGNED32	length;	Measurement Length in bits
UNSIGNED32	start_bit;	

Processor flags

Flag	*Bits*	*Description*	*Init Val*
proc_ena	1	Processing Enable	Default=1
sig_change	1	Significant Change Enable	Default=1
refresh	1	Refresh Request	Default=0
thds_start	1	First THDS Member	Database
thds_member	1	spare	Default=0
last_member	1	Last THDS Member	Database
status_meas	1	Status Measurement	Table Load

ssme_dump	1	SSME Dump Measurement	Database
spare	4		Default=0
tab_type	4	Table type	Table Load

/* Table Type definitions */

Define	*Val*	*Description*
DI_16	1	16 bit discrete parent
DI_64	2	64 bit discrete parent Future
DP_16	3	32 bit digital pattern measurement
DP_64	4	64 bit digital pattern measurement
AN_32	5	32 bit analog measurement
AN_64	6	64 bit analog measurement
TEXT	7	80 byte string Future

Discrete 16 Bit Parent Measurement Specific Table Entry

Data Type	*Item*
UNSIGNED32	proc_ena;
UNSIGNED32	raw_val;
UNSIGNED32	status;
UNSIGNED32	change_ck;
UNSIGNED32	fdid[16];

Digital Pattern 2-32 bits Measurement Specific Table Entry

Data Type	*Item*
UNSIGNED32	raw_val;
UNSIGNED32	proc_val;
UNSIGNED32	mask;
UNSIGNED32	status;

Digital Pattern 33-64 bits Measurement Specific Table Entry

Data Type	*Item*
UNSIGNED32	raw_val_hi;
UNSIGNED32	raw_val_lo;
UNSIGNED32	proc_val_hi;
UNSIGNED32	proc_val_lo;
UNSIGNED32	mask_hi;
UNSIGNED32	mask_lo;
UNSIGNED32	spare;
UNSIGNED32	status;

Analog 32 bit Measurement Specific Table Entry

Data Type	*Item*
UNSIGNED32	raw_val;
FLOAT32	processed_val;
FLOAT32	coef[POLY_ORDER];
UNSIGNED32	status;

Analog 64 bit Measurement Specific Table Entry

Data Type	Item
UNSIGNED32	raw_val_hi;
UNSIGNED32	raw_val_lo;
UNSIGNED32	spare;
UNSIGNED32	status;
FLOAT64	processed_val;

3.3.3.8.1.2 Cross Reference Table

The Cross Reference Table is used to find the measurement descriptor table entry and also the hardware address.

Data Type	*Item*	*Description*
UNSIGNED32	fdid;	Function Designator ID
UNSIGNED32	mdt;	Index into MDT of fdid-parent if child
UNSIGNED32	bit;	Discrete bit number if parent 0xffff
UNSIGNED32	length;	Length in bits of measurement
UNSIGNED32	*hw_ptr;	

All tables are represented a ASCII flat files with one record per line. The first line of each file contains the record count

File name: ***an32.mdt***
Measurement Analog 32bit Record

Field	*Notation*	*Description*
proc_flags	Hex	Processing flags
fdid	Hex	FDID
type	Hex	Measurement Type
sub_type	Hex	Measurement Sub Type
start_bit	Hex	Measurement Start Bit
length	Hex	Measurement Length in bits
coef A0	Decimal	Conversion Coef Single Precision
coef A1	Decimal	Conversion Coef Single Precision
coef A2	Decimal	Conversion Coef Single Precision
coef A3	Decimal	Conversion Coef Single Precision
coef A4	Decimal	Conversion Coef Single Precision
coef A5	Decimal	Conversion Coef Single Precision
data_set_fdid	Hex	FDID of Data Set All zeros if not last member

File name: ***an64.mdt***
Measurement Analog 64bit Record

Field	*Notation*	*Description*
proc_flags	Hex	Processing flags
fdid	Hex	FDID
type	Hex	Measurement Type
sub_type	Hex	Measurement Sub Type
start_bit	Hex	Measurement Length in bits
length	Hex	Measurement Length in bits
data_set_fdid	Hex	FDID of Data Set All zeros if not last member

File Name: ***dp16.mdt***

Measurement Discrete Parent Record

Field	*Notation*	*Description*
proc_flags	Hex	Processing flags
parent_fdid	Hex	FDID-- parent
type	Hex	Measurement Type
sub_type	Hex	Measurement Sub Type
fdid Bit 0	Hex	FDID
fdid Bit 1	Hex	FDID
fdid Bit 2	Hex	FDID
fdid Bit 3	Hex	FDID
fdid Bit 4	Hex	FDID
fdid Bit 5	Hex	FDID
fdid Bit 6	Hex	FDID
fdid Bit 7	Hex	FDID
fdid Bit 8	Hex	FDID
fdid Bit 9	Hex	FDID
fdid Bit 10	Hex	FDID
fdid Bit 11	Hex	FDID
fdid Bit 12	Hex	FDID
fdid Bit 13	Hex	FDID
fdid Bit 14	Hex	FDID
fdid Bit 15	Hex	FDID
data_set_fdid	Hex	FDID of Data Set All zeros if not last member

File Name: ***dp.mdt***

Measurement Digital Pattern File

Field	*Notation*	*Description*
proc_flags	Hex	Processing flags
fdid	Hex	FDID
type	Hex	Measurement Type
sub_type	Hex	Measurement Sub Type
start_bit	Hex	Measurement Length in bits
length	Hex	Parent Length in bits
data_set_fdid	Hex	FDID of Data Set All zeros if not last member

3.3.4 Gateway FEPC Services CSC Test Plan

3.3.4.1 Environment

A development GSE gateway will be connected to a mini-HIM which contains a defined set of I/O cards. At least one card for each of the data types will be present. TCID tables which support the mini-HIM configuration will be present on the GSE gateway local hard drive. Each of the commands supported by the Initialization CSC will be sent. The action taken and the response returned will be verified.

A development gateway will be connected to simulator. At least one of the data types supported will be present. TCID tables which support the gateway configuration will be present on the gateway local hard drive. Tables will be loaded and initialized,Data Acquisition will be activated and the change data output stream will be verified.

3.3.4.2 Test Tools

The GSE gateway will be commanded using the Gateway Integrated Test Environment CSC developed by the gateway group. This tool is capable of generating and displaying the responses of all GSE gateway commands supported.

3.3.4.3 Test Cases

3.3.4.3.1 Common Initialization

1. Load SCID
2. Load TCID
3. Activate
4. Terminate

3.3.4.3.2 Common Measurement Processor

Measurements:
Change data packet formats will be examined for the correct format for the following types

Data Type	*Description*
AM_AU	Analog Unipolar MSB left+1
AM_AB	Analog Bipolar
AM_AHU	Halfword Unsigned
AMF	Analog Measurement Filter
AM_ASM	Bit String Magnitude
AM_AOS	Halfword Overflow Signed
AM_BCD	Binary Coded Decimal
AMDP_AU	Analog Unipolar MSB left+1
AMDP_AB	Analog Bipolar
AMDP_TAC	Tacan Range Word
DM_BD	Binary Discrete
DPM_DEC	Decimal Number
DPM_OCT	Octal Number
DPM_HEX	Hex Number
DPM_BIN	Binary Number
DPM_BCD	Binary Coded Decimal
DPM_TAB	TACAN Bearing Word
MWDP_DEC	Decimal Number
MWDP_OCT	Octal Number

MWDP_HEX	Hex Number
MWDP_BIN	Binary Number
FP_SPL	Float Single Precision
FP_HPL	Float Half Precision
FP_DPL	Float Double Precision
FP_EPL	Float Extended Precision

Commands:
Commands will be issued for the following and responses verified.

Route Code	*Request ID*	*Command*
13	1	Status FDID
13	4	Activate/Inhibit FD Processing
13	5	Activate/Inhibit Processing All
13	6	Read EU Polynomial
13	7	Activate/Inhibit Change Check on FD
13	8	Activate/Inhibit Global Change Check
13	16	Change EU Polynomial

Table Types:
After tables are loaded an entry for each table type will be dumped and examined

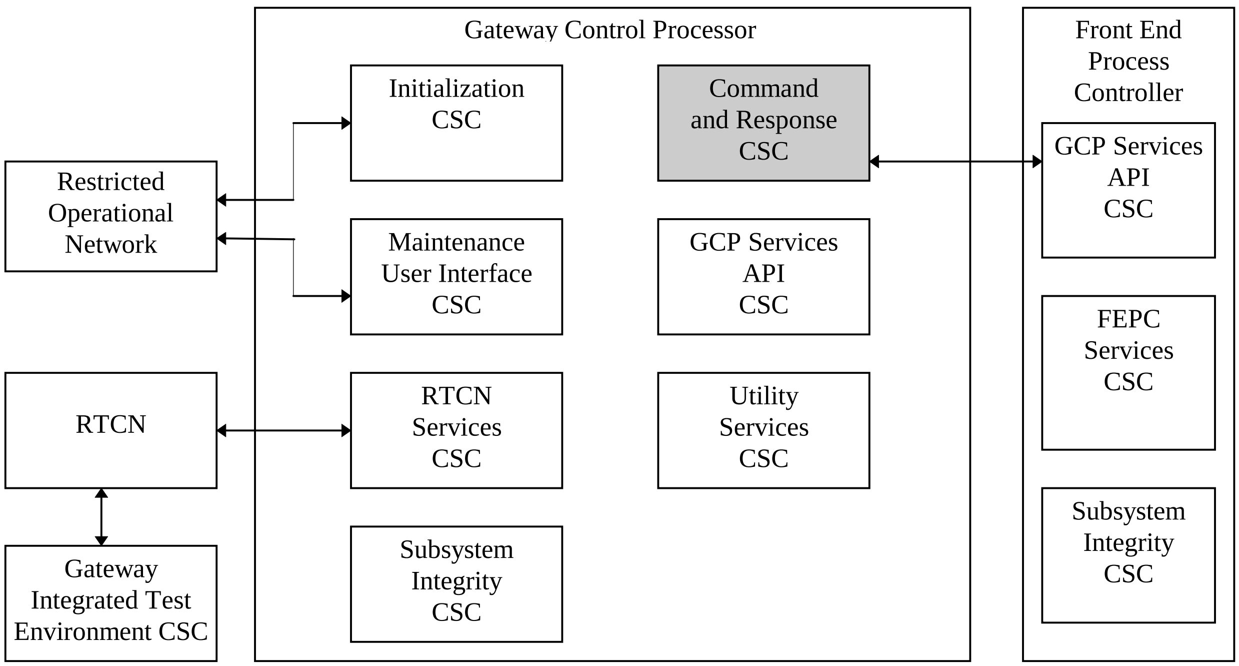
4. Gateway Command and Response CSC

4.1 Gateway Command and Response CSC Introduction

4.1.1 Gateway Command and Response CSC Overview

The Gateway Command and Response CSC is responsible for handling the routing, authentication, and processing of all commands and responses in the Gateway.

Common Gateway Services CSCI



4.1.2 Gateway Command and Response CSC Operational Description

The Gateway Command and Response CSC handles command registration, and command and response processing. During Gateway initialization, each Gateway resident processor registers the Route Code and Request IDs it expects during processing. All shared memory message queues, a routing table, and a transaction table are built using the registered information.

An incoming RTCN command is indexed into the routing table by its Route Code and Request ID, stored in the transaction table if a response is required, checked for command priority, and relayed to the appropriate Gateway processor(s) command queues. A Gateway resident processor's generated response is indexed into the transaction table by transaction ID to verify a response was expected, and relayed to the RTCN.

The Gateway Command and Response CSC is also capable of relaying to the RTCN any commands issued by the Gateway resident processors, and handling the incoming responses associated with those commands.

4.2 Gateway Command and Response CSC Specifications

4.2.1 Gateway Command and Response CSC Groundrules

- Gateway Command and Response CSC will support a Registration shared memory message queue by which each Gateway resident processor may register for specific Command and Response services. The Gateway resident processors will interface to the Registration queue using functions provided by the GCP Services API CSC.
- Gateway Command and Response CSC will support all RTCN using the Gateway RTCN Services CSC.
- Gateway Command and Response CSC will reserve Route Code 0 for internal Gateway communications. No command with a Route Code of 0 will reach a network.
- Gateway Command and Response CSC communication to other Gateway resident processors will be by shared memory message queues. The Gateway resident processors will interface to these queues using functions provided by the GCP Services API CSC.
- Gateway Command and Response CSC will build and maintain the following tables:
  - Route Table: Maps Route Codes and Request IDs to Gateway resident processors.
  - Transaction Table: Tracks commands and their associated required responses.

4.2.2 Gateway Command and Response CSC Functional Requirements

The Functional Requirements for Gateway Command and Response CSC are arranged in the following major/minor functions:

1. Gateway Processor Registration
2. Command Reception
3. Command Generation
4. Response Reception
5. Response Generation

4.2.2.1 Gateway Processor Registration

Gateway Processor Registration is the means by which Common Gateway Services is aware of the Gateway resident processors or GCP resident processes that need Command and Response capabilities.

1. Gateway Command and Response CSC shall provide a Registration shared memory message queue for Gateway resident processor or GCP resident process registration.
2. Gateway Command and Response CSC shall provide the following services to each Gateway resident processor and to GCP resident processes:
   2.1. High and normal priority receive command queues,
   2.2. Generate response queue,
   2.3. Change data queue,
   2.4. High and normal priority generate command queues(<u>limited support in the Thor delivery)</u>,
   2.5. Receive response queue (<u>limited support in the Thor delivery)</u>,
   2.6. Route code and Request ID registration.
3. Registration for any command queues shall result in the creation of both a high and normal priority queues.
4. Gateway Command and Response CSC shall build and maintain a Route Table which will contain registered Route Codes and their associated Request IDs for each Gateway resident processor and GCP resident processes.
5. Route Code and Request ID Registration may occur at any time during Gateway processing.
6. Gateway Command and Response CSC shall build and maintain a Transaction Table which will track responses during command processing.

7. For each requested response reception and command generation queue, Gateway Command and Response CSC shall spawn a task to monitor activity on the queue.

4.2.2.2 Command Reception

Gateway Command Reception is the processing of incoming RTCN commands.

1. Gateway Command and Response shall receive commands asynchronously from the RTCN using the API provided by Network Services CSCI.
2. When an incoming command's Route Code is not found in the Route Table, an Invalid Route Code Response shall be returned to the command's source.
3. When an incoming command's Request ID is not found in the Route Table, an Invalid Request ID Response shall be returned to the command's source.
4. When an incoming command's Route Code and Request ID is found in the Route Table, the command shall be forwarded to each processor/process that registered for that Route Code and Request ID.
5. When an incoming command has been designated in the command header as high priority, the command shall be placed in the appropriate processor(s)/process(es) high priority receive command queue.
6. When an incoming command has been designated in the command header as expecting a response, the command's header shall be copied into the Transaction Table for reference during Response Generation.

4.2.2.3 Command Generation

Gateway Command Generation is the processing and sending of outgoing Gateway commands.

1. Gateway Command and Response shall receive commands asynchronously from the Command Generation shared memory message queues using the GCP Services API CSC.
2. In the Thor Delivery, only Internal Route Code (Route Code 0) commands will be processed.
3. All Internal Route Code commands shall be processed as incoming RTCN commands.
4. When an outgoing command has been designated in the command header as expecting a response, the command's header shall be copied into the Transaction Table for reference during Response Reception.

4.2.2.4 Response Reception

Gateway Response Reception is the processing of incoming responses to Gateway commands.

1. In the Thor Delivery, only responses to internally generated commands will be received.
2. Received Responses shall be routed to the Gateway resident processor that generated the command.

4.2.2.5 Response Generation

Gateway Response Generation is the building and sending of outgoing Responses to the commands processed by the Gateway.

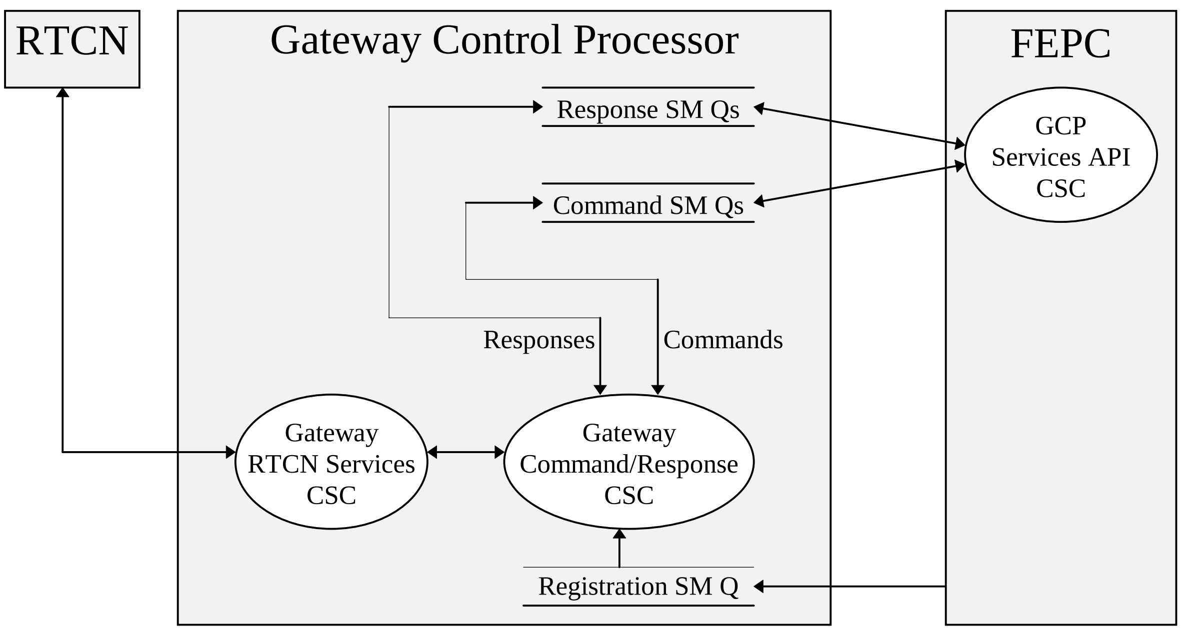
1. Each generate response queue monitor shall read asynchronously from the shared memory message queues.
2. When a generated response is read from the shared memory message queues and not found in the Transaction Table, an Unsolicited Response System Message shall be issued and the Response stored for future access.
3. When a generated response is read from the shared memory message queues and found in the Transaction Table, Gateway Command and Response CSC shall time stamp the packet and write it to the RTCN using the API provided by the Network Services CSCI.
4. When the generated response is one of several expected responses, the response shall be stored in the Transaction Table until all expected responses have been generated. At that time, Gateway Command and Response CSC shall time stamp and send the response to the RTCN.
5. Gateway Command and Response shall support the Command Issued completion code for those commands that require extended periods of time to process.

6. Gateway Command and Response shall support the Continuation bit in the flags field for those commands whose Responses require extremely large payloads.
7. Gateway Command and Response shall support the Command Time-out completion code for those commands that time out while being processed.
8. Gateway Command and Response CSC shall send response packets to the RTCN using the API provided by the Network Services CSCI.
9. All Gateway generated responses will be time stamped immediately prior to transfer to the RTCN. The time stamp will reflect the time the response left the gateway, not when processing occurred.

4.2.3 Gateway Command and Response CSC Performance Requirements

1. Gateway Command and Response CSC shall be capable of processing 500 commands per second.
2. Internal Gateway timeout shall be less than the CCP timeout.

4.2.4 Gateway Command and Response CSC Interfaces Data Flow Diagram



All Gateway resident processors register with the Gateway Command and Response CSC using the Registration shared memory message queue. Registration includes the Route Codes and Request IDs the Gateway resident processor is expecting.

Incoming RTCN Commands are received by the Gateway RTCN Services CSC and forwarded to the Gateway Command and Response CSC. The commands are routed to the Gateway resident processors that registered for that Route Code and Request ID using the Command shared memory message queues. RTCN Responses generated by the Gateway processor are read from the Response shared memory message queues by the Gateway Command and Response CSC. These are then forwarded to the RTCN using the GatewayRTCN Services CSC.

Commands generated by the Gateway resident processor are read from the Command shared memory message queues by the Gateway Command and Response CSC. For Thor, these commands will only be internal commands. These will be routed according to the Request ID of the internal command. Responses received from these commands are received through the Response shared memory message queues.

4.3 Gateway Command and Response CSC Design Specification

The Gateway Command and Response CSC is responsible for routing and tracking Gateway commands and responses. Received RTCN Commands are routed to the appropriate Gateway resident processor through the Command Shared Memory Message Queues. RTCN Responses are received from the Gateway resident processors through Response Shared Memory Message Queues and from this, an RTCN response is generated. Commands and responses are tracked while being relayed to insure their timely delivery.

Gateway Command and Response is comprised of two tables and several tasks. The Registration Task creates and maintains the Registration Shared Memory Queue. This Queue is used by the GCP and the other processors in the Gateway to register for Shared Memory Queue and Route Codes/Request ID Services. When a processor registers, it designates which Route Code/Request ID pairs it wishes to receive, and which Queue services it requires (Command, Response, or Changed Data). The Registration Task is responsible for entering the Route Code/Request ID pairs into the Route Table, and creating the requested Shared Memory Message Queues.

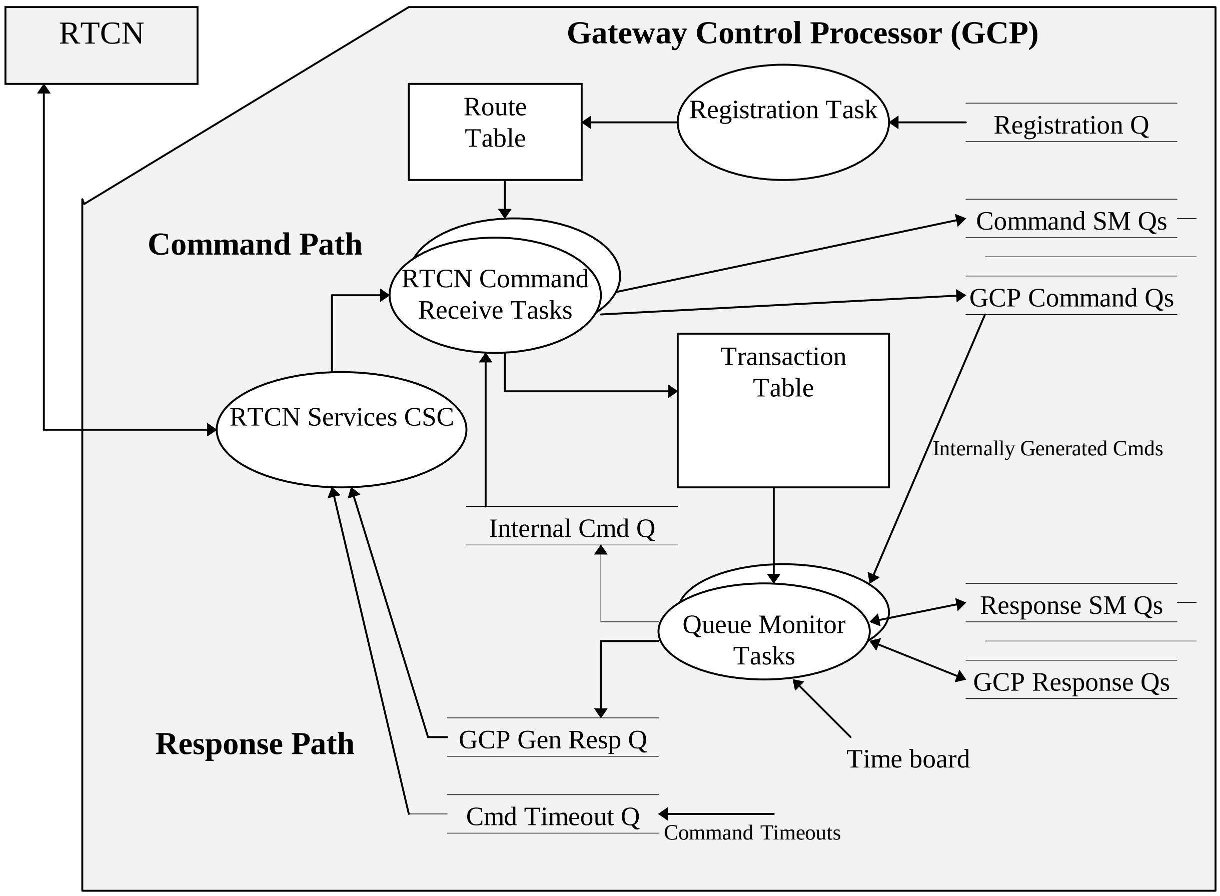
Incoming commands are received from the RTCN Services CSC by one of the RTCN Command Receive Tasks. There is one RTCN Command Receive Task for each incoming multicast data stream communicating with the Gateway. When a command is received, the appropriate RTCN Command Receive Task indexes the Route Table to determine to which Queue(s) to route the incoming command. If a Response to the incoming command is expected, the command is stored in the Transaction Table for future reference, and written to the appropriate command Queue. If multiple Gateway resident processors have registered for the same Route Code/Request ID pair the incoming command will be routed to both processors. The number of processors routed to is also stored in the Transaction Table for use during Response Packet Building.

Responses generated by the Gateway resident processors are received by Gateway Command and Response CSC through the Response Shared Memory Queues. There is an RTCN Response Generation Task for each Response Shared Memory Queue created during registration. The outgoing Response is indexed into the Transaction Table where the original Command is stored. Using the Response and the Command, an RTCN Response is built, time-tagged using the Timer Services CSC, and written to the Response Generation Queue. The RTCN Services CSC is responsible for reading Responses from the Response Generation Queue and forwarding them to the RTCN.

Internally Generated Commands are read from the Command Generation Queues by the Queue Monitor Tasks. These commands are then inserted into the Command Reception path. Like RTCN commands, Internal commands are routed by their Route Code/Request ID pair and, if a Response is expected, stored in the Transaction Table. Responses to the Internal Commands are captured by the Queue Monitor Tasks. The Transaction Table is referenced and the Response is re-routed to the appropriate processor(s) through the Response Queues again.

Command Timeouts are handled through a separate Message Queue. Before an incoming Command is routed to the appropriate processor(s), a timer. The expiration of the timer spawns interrupt code that will send the late Command’s Transaction ID into the Command Timeout Queue. The Error is captured and the Response is built and transmitted with a Command Timeout Completion Code.

4.3.1 Gateway Command and Response CSC Detailed Data Flow



4.3.2 Gateway Command and Response CSC External Interfaces

4.3.2.1 Gateway Command and Response CSC Message Formats

4.3.2.1.1 Error Creating a Shared Memory Object

Message Number =
Message Group = CGS
Severity = Error

Gateway %s: Error creating Shared Memory Object %s.
Gateway Error: %d.

Insert #1 = Text string	Gateway Host Name
Insert #2 = Text string	Shared Memory Object Name
Insert #3 = Integer	Gateway Error Databank Help ID

4.3.2.1.2 Error Accessing a Shared Memory Object

Message Number =

Message Group = CGS
Severity = Error

Gateway %s: Error accessing Shared Memory Object %s.
Gateway Error: %d.

Insert #1 = Text string	Gateway Host Name
Insert #2 = Text string	Shared Memory Object Name
Insert #3 = Integer	Gateway Error Databank Help ID

4.3.2.1.3 Error Accessing the Route Table

Message Number =
Message Group = CGS
Severity = Error

Gateway %s: Error accessing the Route Table.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name
Insert #2 = Integer	Gateway Error Databank Help ID

4.3.2.1.4 Invalid Routing Code

Message Number =
Message Group = CGS
Severity = Informational

Gateway %s: Received an Invalid Route Code.
Route Code: %d.
Source ID: %d.
Destination ID: %d.
Source Transaction ID: %d.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name
Insert #2 = Integer	The Invalid Route Code
Insert #3 = Integer	The Command’s Logical Source ID
Insert #4 = Integer	The Command’s Logical Destination ID
Insert #5 = Integer	The Command’s Transaction ID
Insert #6 = Integer	Gateway Error Databank Help ID

4.3.2.1.5 Invalid Request ID

Message Number =
Message Group = CGS
Severity = Informational

Gateway %s: Received an Invalid Request ID.

Request ID: %d.
Route Code: %d.
Source ID: %d.
Destination ID: %d.
Source Transaction ID: %d.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name
Insert #2 = Integer	The Invalid Request ID
Insert #3 = Integer	The Command's Route Code
Insert #4 = Integer	The Command's Logical Source ID
Insert #5 = Integer	The Command's Logical Destination ID
Insert #6 = Integer	The Command's Transaction ID
Insert #7 = Integer	Gateway Error Databank Help ID

4.3.2.1.6 Error Adding a Transaction Entry

Message Number =
Message Group = CGS
Severity = Error

Gateway %s: Error adding a Transaction Table Entry.
Source ID: %d.
Route Code: %d.
Request ID: %d.
Destination ID: %d.
Source Transaction ID: %d.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name
Insert #2 = Integer	The Command's Logical Source ID
Insert #3 = Integer	The Command's Route Code
Insert #4 = Integer	The Command's Request ID
Insert #5 = Integer	The Command's Logical Destination ID
Insert #6 = Integer	The Command's Transaction ID
Insert #7 = Integer	Gateway Error Databank Help ID

4.3.2.1.7 Error Removing a Transaction Entry

Message Number =
Message Group = CGS
Severity = Error

Gateway %s: Error removing a Transaction Table Entry.
Gateway Transaction Index: %d.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name

Insert #2 = Integer	The Command's Local Transaction Index
Insert #3 = Integer	Gateway Error Databank Help ID

4.3.2.1.8 Error Accessing the Transaction Table

Message Number =
Message Group = CGS
Severity = Error

Gateway %s: Error accessing the Transaction Table.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name
Insert #2 = Integer	Gateway Error Databank Help ID

4.3.2.1.9 Error Creating a Transaction Timer

Message Number =
Message Group = CGS
Severity = Error

Gateway %s: Error creating a Transaction Timer.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name
Insert #2 = Integer	Gateway Error Databank Help ID

4.3.2.1.10 Timeout Error

Message Number =
Message Group = CGS
Severity = Informational

Gateway %s: Command timed out.
Source ID: %d.
Route Code: %d.
Request ID: %d.
Destination ID: %d.
Source Transaction ID: %d.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name
Insert #2 = Integer	The Command's Logical Source ID
Insert #3 = Integer	The Command's Route Code
Insert #4 = Integer	The Command's Request ID
Insert #5 = Integer	The Command's Logical Destination ID
Insert #6 = Integer	The Command's Transaction ID
Insert #7 = Integer	Gateway Error Databank Help ID

4.3.2.1.11 Unsolicited FEPC Response

Message Number =
Message Group = CGS
Severity = Error

Gateway %s: FEPC generated an unsolicited response.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name
Insert #7 = Integer	Gateway Error Databank Help ID

4.3.2.1.12 Error Sending to a Shared Memory Message Queue

Message Number =
Message Group = CGS
Severity = Error

Gateway %s: Error sending to a Shared Memory Message Queue.
Queue ID: %d.
Queue Name: %s.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name
Insert #2 = Integer	The VxWorks Queue ID
Insert #3 = Text string	The Queue’s ASCII Name
Insert #4 = Integer	Gateway Error Databank Help ID

4.3.2.1.13 Error Reading from a Shared Memory Message Queue

Message Number =
Message Group = CGS
Severity = Error

Gateway %s: Error reading from a Shared Memory Message Queue.
Queue ID: %d.
Queue Name: %s.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name
Insert #2 = Integer	The VxWorks Queue ID
Insert #3 = Text string	The Queue’s ASCII Name
Insert #4 = Integer	Gateway Error Databank Help ID

4.3.2.2 Gateway Command and Response CSC Display Formats

None

4.3.2.3 Gateway Command and Response CSC Input Formats

None

4.3.2.4 Gateway Command and Response CSC Recorded Data

None

4.3.2.5 Gateway Command and Response CSC Printer Formats

None

4.3.2.6 Gateway Command and Response CSC Interprocess Communications

All Gateway Interprocess Communications are all C-C commands and responses intended for and generated by the Gateway. The details of these C-C formats are outlined in the Thor Packet Payload document. All C-C commands are handled, but not manipulated by the Gateway Command and Response CSC.

4.3.2.7 Gateway Command and Response CSC External Interface Calls

4.3.2.7.1 Table Analysis Functions

4.3.2.7.1.1 Print Route Table

This function is useful for verifying the routing of incoming commands.

void cgsPrintRouteTable( );

Description: Outputs the Route Table to the Console port.

Parameters: None
Returns: None

4.3.2.7.1.2 Print Transaction Table

This function is useful for verifying the current state of outstanding Gateway transactions.

void cgsPrintTransactionTable( );

Description: Outputs the Transaction Table to the Console port.

Parameters: None
Returns: None

4.3.2.8 Gateway Command and Response CSC Table Formats

4.3.2.8.1 Gateway Route Table

The Gateway Route Table is a Table that is resident on the Gateway Control Processor and controls the Routing of all incoming Gateway commands. As Gateway resident processors register through the Registration Queue, the Route Codes, Request IDs, and priorities that are registered are stored and tracked in the Gateway Route Table. As commands come into the Gateway, this table is referenced and commands are forwarded to the appropriate shared memory message queues.

<Route Code> <Request ID> <Route Name> <Priority> <Num CPUs> <Route Target Ptr>

Route Code: The Route Code that was registered.
Request ID: The Request ID that was registered.
Route Name: The ASCII index into the Hash Table. This is an ASCII concatenation of the Route Code and Request ID.
Priority: The Priority is an indicator of the urgency of the expected command. If this flag is set during Registration, this command will always be sent to the High Priority Command Queue.
Num CPUs: The number of Gateway resident processors that have registered for this Route Code/Request ID pair.
Route Target Ptr: A pointer to the linked list of structures which contain information for each of the Gateway resident processors that have registered for this Route Code/Request ID pair. Each Route Target structure contains the following information:

- CPU ID: The Gateway CPU ID for the processor registered for this Route Code/Request ID pair.
- CPU Name: The registered name of the CPU (e.g. “cpu01”).
- Hi Cmd Q ID: The VxWorks message queue ID of this Gateway resident processor’s High Priority Command Queue.
- Norm Cmd Q ID: The VxWorks message queue ID of this Gateway resident processor’s Normal Priority Command Queue.
- Cmd Ready Sem: The VxWorks semaphore ID of this Gateway resident processor’s Command Ready Semaphore.

4.3.2.8.2 Gateway Transaction Table

The Gateway Transaction Table is a Table that is resident on the Gateway Control Processor and tracks the incoming/outgoing commands and their associated responses. If a response is requested of an incoming command, that command’s header information is stored in the Transaction Table. Also, a timer is started as the command is passed to the FEPC. If the command is responded to, the timer is cancelled and the response is transmitted. If a response is late, the timer expires and a Command Timeout response is returned to the sender.

<Trans Hdr> <Resp Ptr>

Trans Hdr: The header information for each Transaction Table Entry. This is a structure which contains the following information:

- Trans ID: The local Gateway Transaction Index (NOTE: This is not the Transaction ID of the stored command)
- Task Name: The name given the task when it is created
- Num CPUs: The Number of CPUs to which the command was distributed
- Sender: The Multicast Stream ID (socket ID) upon which the command was received
- Timer ID: The ID of the Timer tracking the command
- Issued Cnt: The number of COMMAND ISSUED responses that have been received
- Complete Cnt: The number of non-COMMAND ISSUED responses that have been received
- Max Timeout: The maximum timeout for this command
- Cmd Hdr: A copy of the command’s header

Resp Ptr: A pointer to the linked list of structures which contain response information from each of the Gateway resident processors to which the commands were sent. Each Transaction Response structure contains the following information:

- Resp CPU ID: The responder’s local Gateway CPU ID
- Issued Resp: A copy of the CPU’s COMMAND ISSUED response
- Comp Resp: A copy of the CPU’s non-COMMAND ISSUED response

4.3.3 Gateway Command and Response CSC Test Plan

4.3.3.1 Environment

The Common Gateway Services CIT will be performed in conjunction with the GSE Gateway Services CIT. A development Gateway will be configured as a GSE Gateway. RTCN commands will be sent to the development Gateway. The action taken and the responses returned will be verified.

4.3.3.2 Test Tools

The Gateway Integrated Test Environment CSC will be used to test the Gateway Command and Response CSC.

4.3.3.3 Test Cases

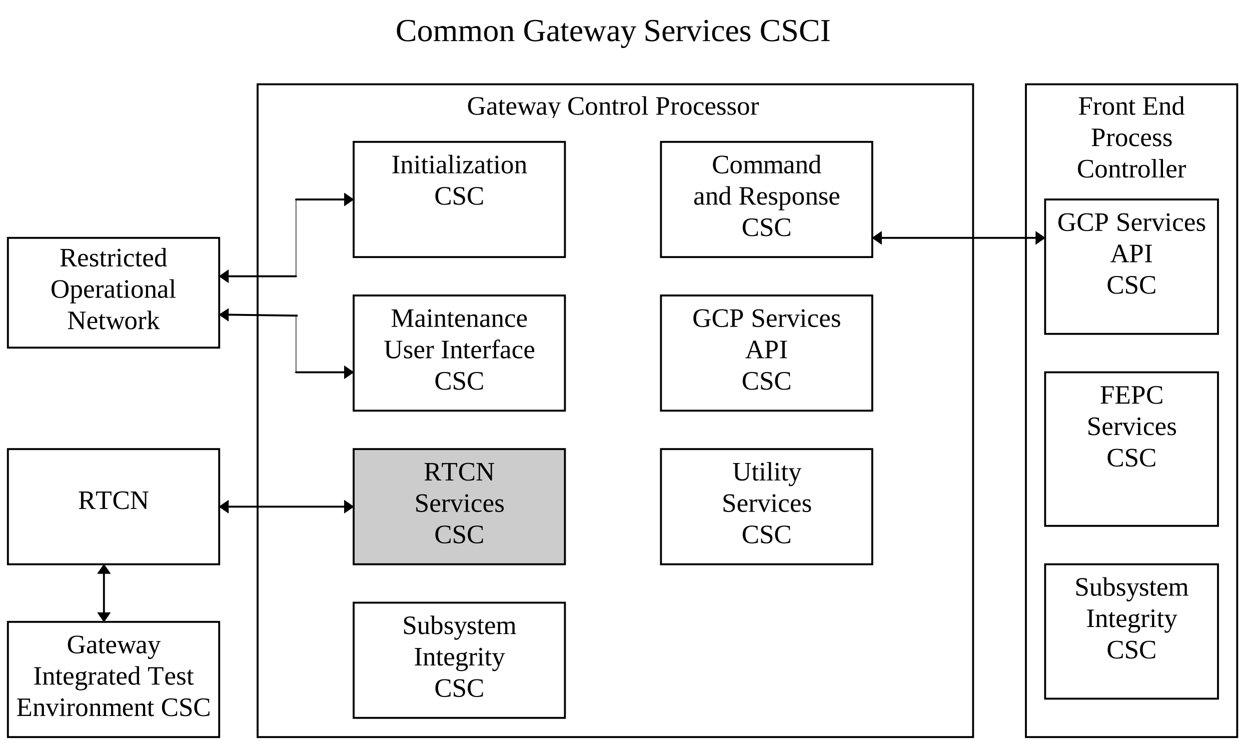
1. Commanding the Gateway with an expected successful response.
2. Commanding the Gateway with an invalid Route Code.
3. Commanding the Gateway with an invalid Request ID.

5. Gateway RTCN Services CSC

5.1 Gateway RTCN Services CSC Introduction

5.1.1 Gateway RTCN Services CSC Overview

The Gateway RTCN Services CSC is responsible for the different services needed by the Gateway when interfacing with the RTCN. It is also responsible for output of Change Data packets. It is part of the Common Gateway Services CSCI and is resident in the GCP.



5.1.2 Gateway RTCN Services CSC Operational Description

The Gateway RTCN Services CSC contains the API provided by the System Services CSCI which is used by the Gateway components when interfacing with the RTCN. Also, when interrupted by the Time board, the Gateway RTCN Services CSC will build RTCN change data packets and transfer them to the RTCN. These RTCN change data packets contain change measurements provided by the Gateway.

5.2 Gateway RTCN Services CSC Specifications

5.2.1 Gateway RTCN Services CSC Groundrules

- The System Services CSCI API library used by the Gateway RTCN Services CSC for all communications over the RTCN will be linked as part of the Gateway SCID.
- The Gateway Time board will interrupt RTCN Services CSC at the system synchronous rate.

- All Gateway resident processors will provide change measurements using the Change Data Shared Memory Message Queue.

5.2.2 Gateway RTCN Services CSC Functional Requirements

The Gateway RTCN Services CSC shall provide change data packets to the RTCN at the system synchronous rate.

The Functional Requirements for the Gateway RTCN Services CSC are arranged in the following major functions:

1. RTCN Interface Functions
2. Change Data Generation

5.2.2.1 RTCN Interface Functions

The RTCN Interface Functions use the Network Services CSCI API to aid other Gateway CSCs in the management of network data streams.

1. Gateway RTCN Services CSC shall provide functions to open and close Gateway network data streams.
2. Gateway RTCN Services shall maintain a Network Streams Table to track the state and activity of each network stream.

5.2.2.2 Change Data Generation

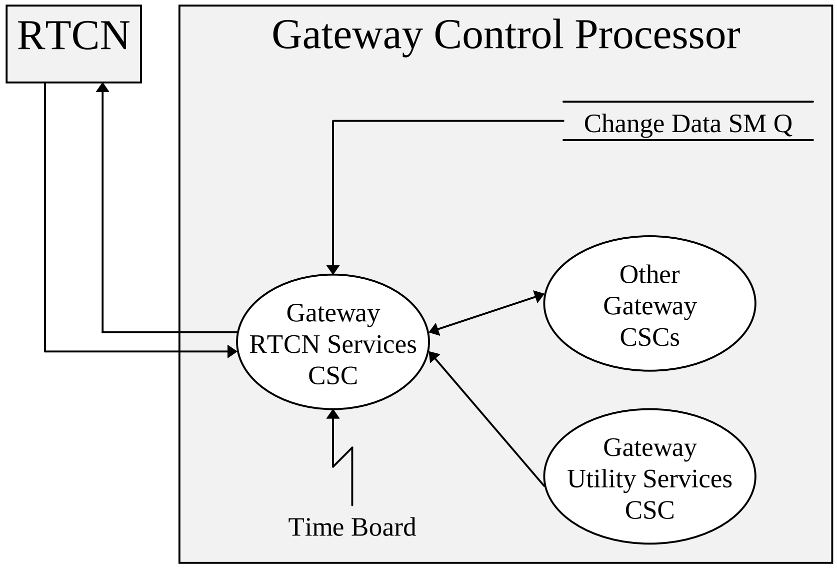
The Change Data Generation functions are responsible for the output of Change Data from the Gateway.

1. Gateway RTCN Services CSC shall receive Change Data entries asynchronously from the Change Data Shared Memory Message Queues.
2. Gateway RTCN Services CSC shall timestamp all received Change Data entries, and add them to the Gateway's outgoing Change Data packet.
3. When necessary, Gateway RTCN Services CSC shall include a millisecond offset time entry into the Change Data packet preceding the appropriate Change Data entry.
4. Gateway RTCN Services CSC shall transmit Change Data packets to the RTCN at the System Synchronous Rate.

5.2.3 Gateway RTCN Services CSC Performance Requirements

1. The Gateway RTCN Services CSC shall provide change data packets to the RTCN at the system synchronous rate.

5.2.4 Gateway RTCN Services CSC Interfaces Data Flow Diagram



Gateway RTCN Services CSC is spawned by the Gateway Initialization CSC.

All communications with the RTCN are handled by the Gateway RTCN Services CSC. Commands and Responses are channeled to the Gateway Command and Response CSC. System Messages and System Events that are generated by the Utility Services CSC are also handled.

Change Data is read from the Change Data Shared memory Message Queue, stored in a packet and sent at the System Synchronous Rate. The System Synchronous Rate is provided by Gateway Time board.

5.3 Gateway RTCN Services CSC Design Specification

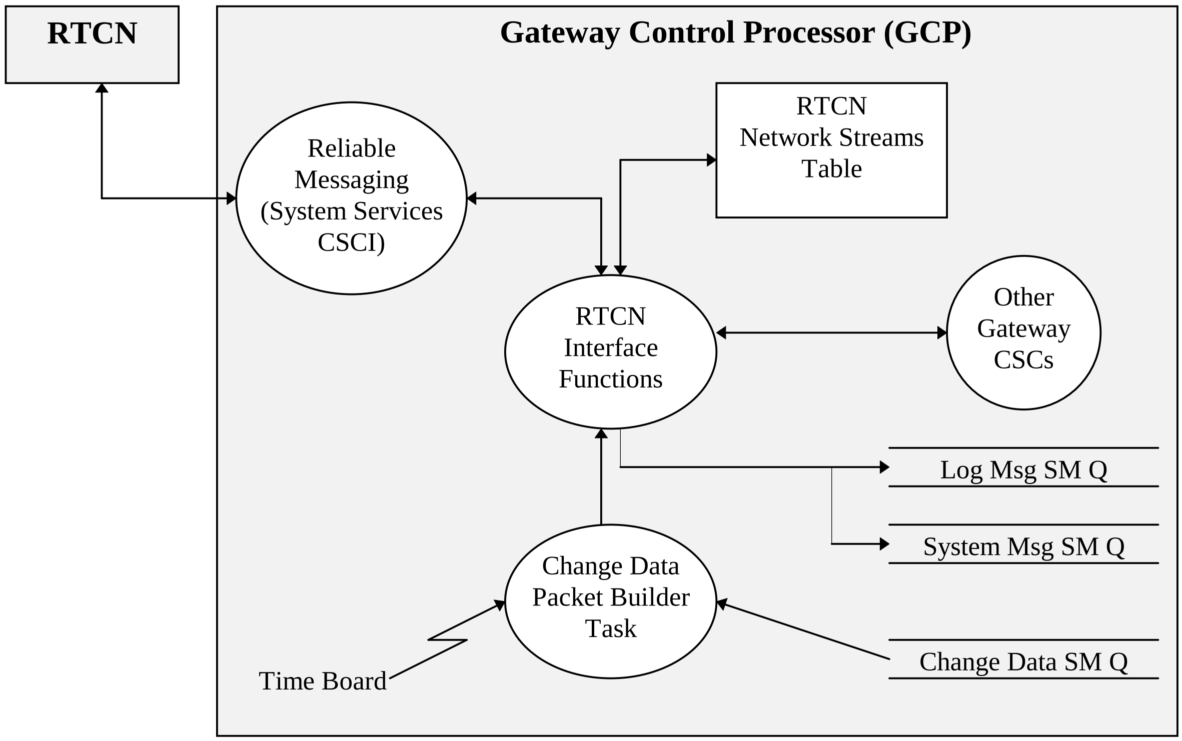
The Gateway RTCN Services CSC contains the APIs provided by the System Services CSCI which is used by the Gateway components when interfacing with the RTCN. The Gateway RTCN Services CSC also provides the Change Data Packet Builder which builds and transmits the change data via RTCN. The changed data is provided by the FEPC. This transmission is done at the System Synchronous Rate.

Much of the RTCN Services CSC is a layer of software over the System Services CSCI's Reliable Messaging API calls. A version of Reliable Messaging is resident on the Gateway Control Processor. The RTCN Services CSC layer of software provides the Gateway with the capability to track and control a dynamic version of the Reliable Messaging Multicast Streams file. This is stored and maintained as the RTCN Network Streans Table. This is a Table resident in memory and contains all information about all RTCN Stream connections. All other Gateway CSCs will have access to the information in the RTCN Network Streams Table using the External Interface calls provided in this document (see 5.3.2.7). All Multicast stream open, close, send, and receive calls will be using the functions provided by the RTCN Services CSC.

The Changed Data Packet Builder task provides the Change Data packets at the System Synchronous Rate. This task's intricate tming is supported by an interrupt from the Time Board at the System Synchronous Rate. This task reads Change Data entries from the Change Data Shared Memory Message Queue located on the Gateway Control Processor. Each collected entry is added to the Change Data packet. When the interrupt is received from the Time Board, the packet is transmitted to the RTCN using RTCN Services CSC's function calls. The Changed Data Packet

Builder task is initialized/terminated by the Gateway Initialization CSC. Errors and System Messages are passed to Utility Services CSC via the Log Message and System Message Shared Memory Message Queues.

5.3.1 Gateway RTCN Services CSC Detailed Data Flow



5.3.2 Gateway RTCN Services CSC External Interfaces

5.3.2.1 Gateway RTCN Services CSC Message Formats

5.3.2.1.1 Error Opening Multicast Data Stream

Message Number =
Message Group = CGS
Severity = Error

Gateway %s: Multicast Stream %s could not be opened.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name
Insert #2 = Text string	Multicast Stream Name
Insert #3 = Integer	Gateway Error Databank Help ID

5.3.2.1.2 Error Closing Multicast Data Stream

Message Number =
Message Group = CGS

Severity = Error

Gateway %s: Multicast Stream %s on ID %d could not be closed.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name
Insert #2 = Text string	Multicast Stream Name
Insert #3 = Integer	Multicast Stream Identifier (socket ID)
Insert #4 = Integer	Gateway Error Databank Help ID

5.3.2.1.3 Error Sending to a Multicast Data Stream

Message Number =
Message Group = CGS
Severity = Error

Gateway %s: Error sending to Multicast Stream %s on ID %d.
CLM Error %d: %s.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name
Insert #2 = Text string	Multicast Stream Name
Insert #3 = Integer	Multicast Stream Identifier (socket ID)
Insert #4 = Integer	ConnectionLess Messaging (CLM) Error Number
Insert #5 = Text string	ConnectionLess Messaging (CLM) Error Description
Insert #6 = Integer	Gateway Error Databank Help ID

5.3.2.1.4 Error Receiving from a Multicast Data Stream

Message Number =
Message Group = CGS
Severity = Error

Gateway %s: Error receiving from Multicast Stream %s on ID %d.
CLM Error %d: %s.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name
Insert #2 = Text string	Multicast Stream Name
Insert #3 = Integer	Multicast Stream Identifier (socket ID)
Insert #4 = Integer	ConnectionLess Messaging (CLM) Error Number
Insert #5 = Text string	ConnectionLess Messaging (CLM) Error Description
Insert #6 = Integer	Gateway Error Databank Help ID

5.3.2.2 Gateway RTCN Services CSC Display Formats

None

5.3.2.3 Gateway RTCN Services CSC Input Formats

None

5.3.2.4 Gateway RTCN Services CSC Recorded Data

None

5.3.2.5 Gateway RTCN Services CSC Printer Formats

None

5.3.2.6 Gateway RTCN Services CSC Interprocess Communications

None

5.3.2.7 Gateway RTCN Services CSC External Interface Calls

5.3.2.7.1 Open RTCN Multicast Stream

```
STATUS cgsOpenRtcnStream(   char    *stream_name,
                            int     stream_type  )
```

Parameters: stream_name — The name of the stream as found in the Reliable Multicast Table.

stream_type — The stream type (Rel_MC_Send, Rel_MC_Recv, etc.).

Returns: OK or ERROR as defined in vxWorks.h

Description: Opens an RTCN Stream, and creates an entry in the RTCN Network Streams Table to track the stream. If an error occurs during this function call, System Message #(TBD) (See 5.3.2.1.1) will be outputted.

5.3.2.7.2 Close RTCN Multicast Stream

```
STATUS cgsCloseRtcnStream( char    *stream_name )
```

Parameters: stream_name — The name of the stream as found in the Reliable Multicast Table.

Returns: OK or ERROR as defined in vxWorks.h

Description: Closes the specified RTCN Stream, and removes its entry in the RTCN Network Streams Table. If an error occurs during this function call, System Message #(TBD) (See 5.3.2.1.2) will be outputted.

5.3.2.7.3 Get Multicast Stream ID

```
int    cgsGetStreamId( char    *stream_name )
```

Parameters: stream_name — The name of the stream as found in the Reliable Multicast Table.

Returns: The Stream ID or -1 if an error occurs

Description: Accesses the RTCN Network Streams Table and returns the ID for the Stream Name specified in the argument.

5.3.2.7.4 Send to an RTCN Multicast Stream

```
int cgsSendToRtcnStream( int    stream_id,
                         void   *send_buffer_ptr,
                         int    send_buffer_size )
```

Parameters: stream_id — The Stream ID (socket ID) of the stream as found in the RTCN Network Streams Table.

send_buffer_ptr — A pointer to the buffer to be sent on the stream.

send_buffer_size — The size in bytes of the buffer to be sent on the stream.

Returns: The number of bytes sent, or -1 on an error.

Description: Sends the data pointed to by send_buffer_ptr to the specified RTCN Stream, and updates any metrics associted with the RTCN Network Streams Table. If an error occurs during this function call, System Message #(TBD) (See 5.3.2.1.3) will be outputted.

5.3.2.7.5 Receive from an RTCN Multicast Stream

```
int cgsRecvFromRtcnStream(  int    stream_id,
                            void   *recv_buffer_ptr,
                            int    max_buffer_size )
```

Parameters: stream_id — The Stream ID (socket ID) of the stream as found in the RTCN Network Streams Table.

recv_buffer_ptr — A pointer to the buffer where the received data will be stored.

max_buffer_size — The maximum size in bytes of the buffer to be read from the stream.

Returns: The number of bytes received, or -1on an error.

Description: Receives data the specified RTCN Stream, stores the data in the buffer pointed to by recv_buffer_ptr, and updates any metrics associted with the RTCN Network Streams Table. If an error occurs during this function call, System Message #(TBD) (See 5.3.2.1.4) will be outputted.

5.3.2.7.6 Write Change Data (as defined in Gateway GCP Services API CSC)

5.3.2.8 Gateway RTCN Services CSC Table Formats

5.3.2.8.1 Reliable Multicast Table

Each communications path to/from the Gateway is named in this file and a mapping from this name to the multicast address/port number for data and acknowledgments associated with this path is provided. The proposed format for each entry is as follows:

<Stream name> <Address> <Port> <Timeout> <Num retries>

where:
Stream name — The ASCII name of the data stream
Address — The IP address of the data stream (broadcast or multicast)

Port	The UDP port number associated with the data stream
Timeout	The Timeout between transmission retries in milliseconds
Num retries	The maximum number of retries before returning an error

This table is provided by System Services CSCI. It must be present on the Gateway Disk in the /sd0:/scid directory.

5.3.2.8.2 RTCN Network Streams Table

The RTCN Network Streams Table is a version of the Reliable Multicast Table stored in the Gateway's memory. It also contains additional counters and status information. The format of this table is as follows:

<Stream name> <Stream ID> <Active> <Error Cnt> <Use Cnt> <Address> <Port> <Timeout> <Num retries>

where:

Stream name	The ASCII name of the data stream
Stream ID	The numerical Stream ID (socket ID) of the data stream
Active	A Boolean which is true when the Stream is successfully transmitting
Error Cnt	The number of Errors which have occurred using this data stream
Use Cnt	A counter indicating the number of packets transmitted on this data stream
Address	The IP address of the data stream (broadcast or multicast)
Port	The UDP port number associated with the data stream
Timeout	The Timeout between transmission retries in milliseconds
Num retries	The maximum number of retries before returning an error

5.3.3 Gateway RTCN Services CSC Test Plan

5.3.3.1 Environment

The Common Gateway Services CIT will be performed in conjunction with the GSE Gateway Services CIT. A development Gateway will be configured as a GSE Gateway. RTCN commands will be sent to the development Gateway. The action taken and the responses returned will be verified.

5.3.3.2 Test Tools

The Gateway Integrated Test Environment CSC will be used to test the Gateway RTCN Services CSC.

5.3.3.3 Test Cases

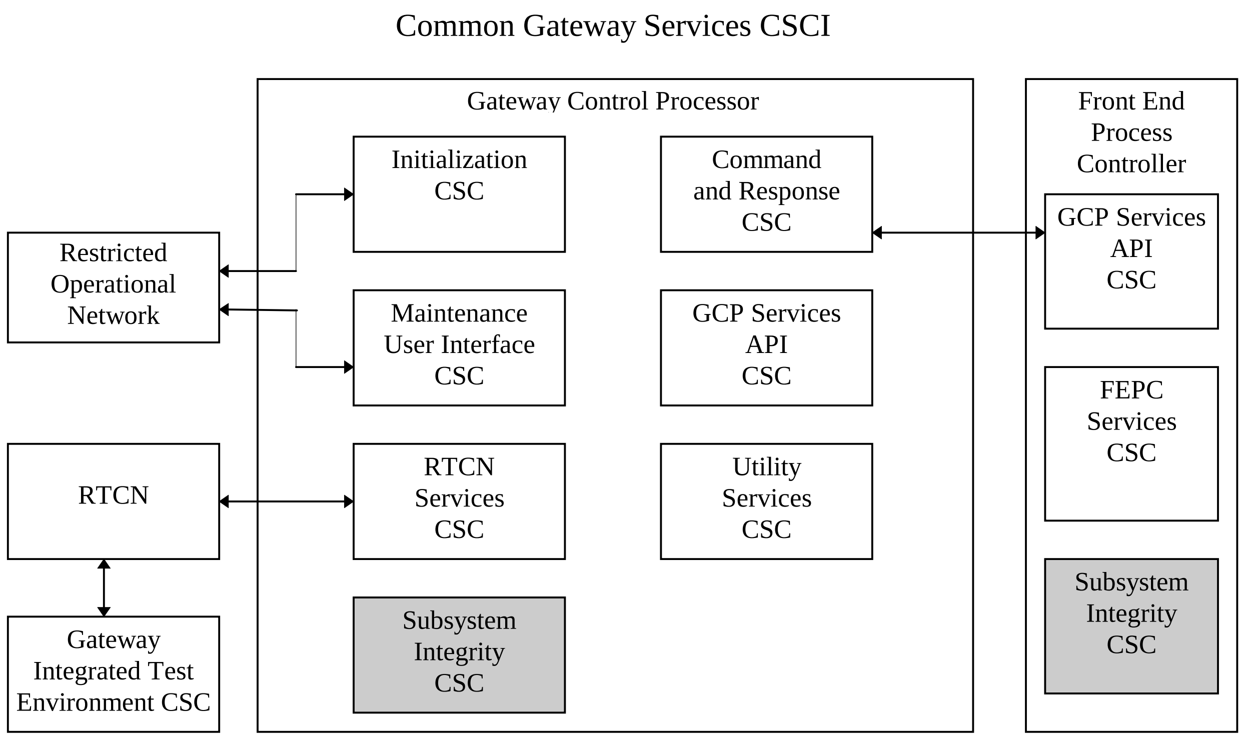
1. Verify the capability to communicate over the RTCN.
2. Verify that the Gateway is generating changed data packets at the system synchronous rate using the RTCN Analyzer.
3. Exercise RTCN Services CSCs external functions. Check for outgoing System messages using the RTCN Analyzer.

6. Gateway Subsystem Integrity CSC

6.1 Gateway Subsystem Integrity CSC Introduction

6.1.1 Gateway Subsystem Integrity CSC Overview

The Gateway Subsystem Integrity CSC is responsible for maintaining task and board level integrity in the Gateway.



6.1.2 Gateway Subsystem Integrity CSC Operational Description

The Gateway Subsystem Integrity CSC is a set of tasks and tables that monitors the health of all tasks in the Gateway. Also, this CSC builds and maintains a Shared Memory area through which each CPU in the Gateway may update Health and Status information. This information is tracked closely by the Gateway Control Processor (GCP). A mechanism is provided by this CSC to automatically report Health and Status FDs on a cyclic basis. This CSC is responsible for the generation and reception of System Event codes. Gateway Subsystem Integrity is also responsible for recovery/termination in the case of a Gateway Health problem.

6.2 Gateway Subsystem Integrity CSC Specifications

6.2.1 Gateway Subsystem Integrity CSC Groundrules

- All Gateway Subsystem Integrity changed data will be written to the Change Data shared memory message queue. From there, Gateway RTCN Services CSC is responsible for including them as part of the Gateway Change Data output stream.
- Gateway task monitoring will not apply to VxWorks Operating System tasks (possible in a future release).

- Every monitored task in the Gateway will be assumed a Critical task unless that task explicitly declares itself as Non-critical.
- Test Build CSCI will provide each Gateway with tables correlating Subsystem Integrity FD Names to the FDIDs for each physical Gateway.
- An independent RTCN network stream will be provided by System Services CSCI for System Event codes.

6.2.2 Gateway Subsystem Integrity CSC Functional Requirements

The Functional Requirements for the Gateway Subsystem Integrity CSC are arranged in the following major functions:

1. Gateway Task Health and Status
2. Health and Status Function Designators
3. Gateway Processor Integrity
4. System Event codes

6.2.2.1 Gateway Task Health and Status

Gateway Task Health and Status is a set of task monitoring functions that will reside on every processor in the Gateway.

1. Gateway Subsystem Integrity shall build and maintain a Task Health Table on each processor in the Gateway which will store task Health and Status information.
2. Gateway Subsystem Integrity shall provide a Task registration function which will automatically reserve an entry in the Task Health Table whenever a task is created. This function will be called by the Initialization CSC for that processor (Gateway Initialization CSC, or Gateway FEPC Services CSC).
3. Gateway Subsystem Integrity shall provide a Task un-registration function which will automatically delete an entry from the Task Health Table whenever a task is deleted. This function will be called by the Gateway Initialization CSC.
4. Gateway Subsystem Integrity shall update all of a Gateway Processor’s task entries in its Task Health Table (TBD) update rate period.
5. Gateway Subsystem Integrity shall verify the existence of all tasks every (TBD) update rate period.
6. Gateway Subsystem Integrity shall check the processing state of all tasks every (TBD) update rate period.
7. If a Critical task is found to be absent or suspended, Gateway Subsystem Integrity shall discontinue processing, attempt to send a System Event message, attempt to send a System Message, and perform Recovery.
8. If a Non-critical task is found to be absent or suspended, Gateway Subsystem Integrity shall send a System Message.

6.2.2.2 Cyclic Health and Status Function Designators

Cyclic Health and Status Function Designators are those FDs which require periodic updates. Gateway Subsystem Integrity CSC will provide a generic means of reporting these reliably.

1. Gateway Subsystem Integrity shall provide a registration function to allow other CSC's to register the address, size, and FD name for health/status FD's.
2. Gateway Subsystem Integrity shall maintain a list of health/status measurements and their associated FDID's.
3. Gateway Subsystem Integrity will scan this list and output all data which has changed at a (TBD) update rate.
4. Gateway Subsystem Integrity CSC shall provide the capability to output Cyclic Health and Status FD information on demand.

6.2.2.3 Gateway Processor Integrity

Gateway Processor Integrity consists of a Task and Shared Memory Area residing on the Gateway Control Processor (GCP). The Shared Memory Area contains a Health counter for each processor in the Gateway. It is the responsibility of each Gateway resident processor to update its count in this Shared Memory Area, and to verify the counts of the other processors in the Gateway.

1. Gateway Subsystem Integrity CSC shall build and maintain a Gateway Health Shared Memory Area on the GCP.
2. Gateway Subsystem Integrity shall update the Health counter for each Gateway processor upon which it resides.
3. If the GCP fails to update its Health count, the FEPC shall discontinue processing, command the GCP to discontinue processing, and perform Recovery.
4. If the FEPC fails to update its Health count, the GCP shall command the FEPC to discontinue processing, attempt to send a System Event message, attempt to send a System Message, and perform Recovery.
5. If all counts have been properly updated, Gateway Subsystem Integrity shall send a Gateway wide Health count to the Change Data shared memory message queue at a (TBD) rate.

6.2.2.4 System Event codes

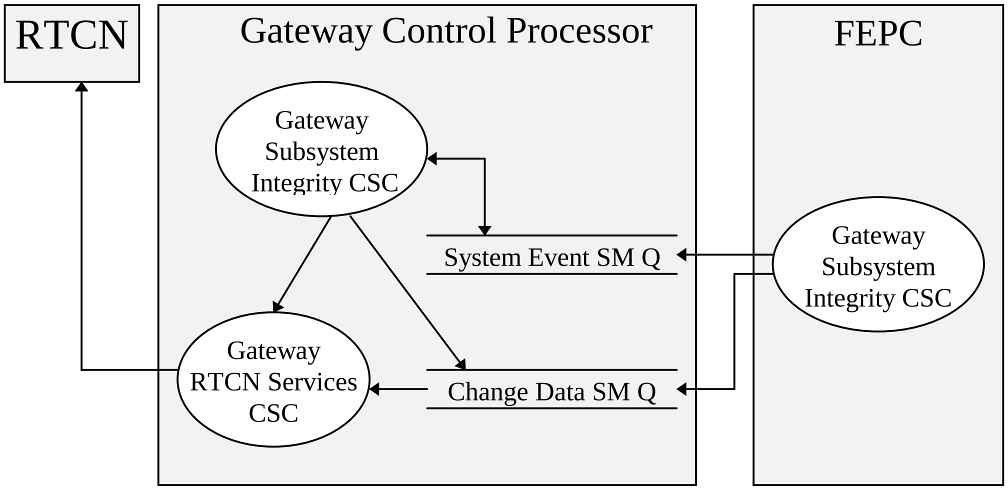
The System Event routines provide all processes on the Gateway the ability to send and receive System Event codes.

1. Gateway Subsystem Integrity CSC shall provide an independent shared memory message queue and handling task for System Event processing.
2. Gateway Subsystem Integrity CSC shall provide a method for all resources on the Gateway to send System Event codes.
3. Under the following conditions the Gateway Subsystem Integrity CSC shall transmit the terminal System Event code:
   - 3.1. The Gateway Control Processor fails to update its Health Counter.
   - 3.2. The Front-End Process Controller fails to update its Health Counter.
   - 3.3. A Critical task on the Gateway is found to be absent or suspended.
   - 3.4. The FEPC requests a System Event code to be transmitted (using the shared memory queue).

6.2.3 Gateway Subsystem Integrity CSC Performance Requirements

1. Gateway Subsystem Integrity shall send Gateway Subsystem Integrity change data at a (TBD) rate.

6.2.4 Gateway Subsystem Integrity CSC Interfaces Data Flow Diagram



Gateway Subsystem Integrity CSC is spawned by the Gateway Initialization CSC. All Subsystem Integrity information is channeled through the Change Data Shared Memory Message Queue. From there, it is routed to the RTCN using the Gateway RTCN Services CSC.

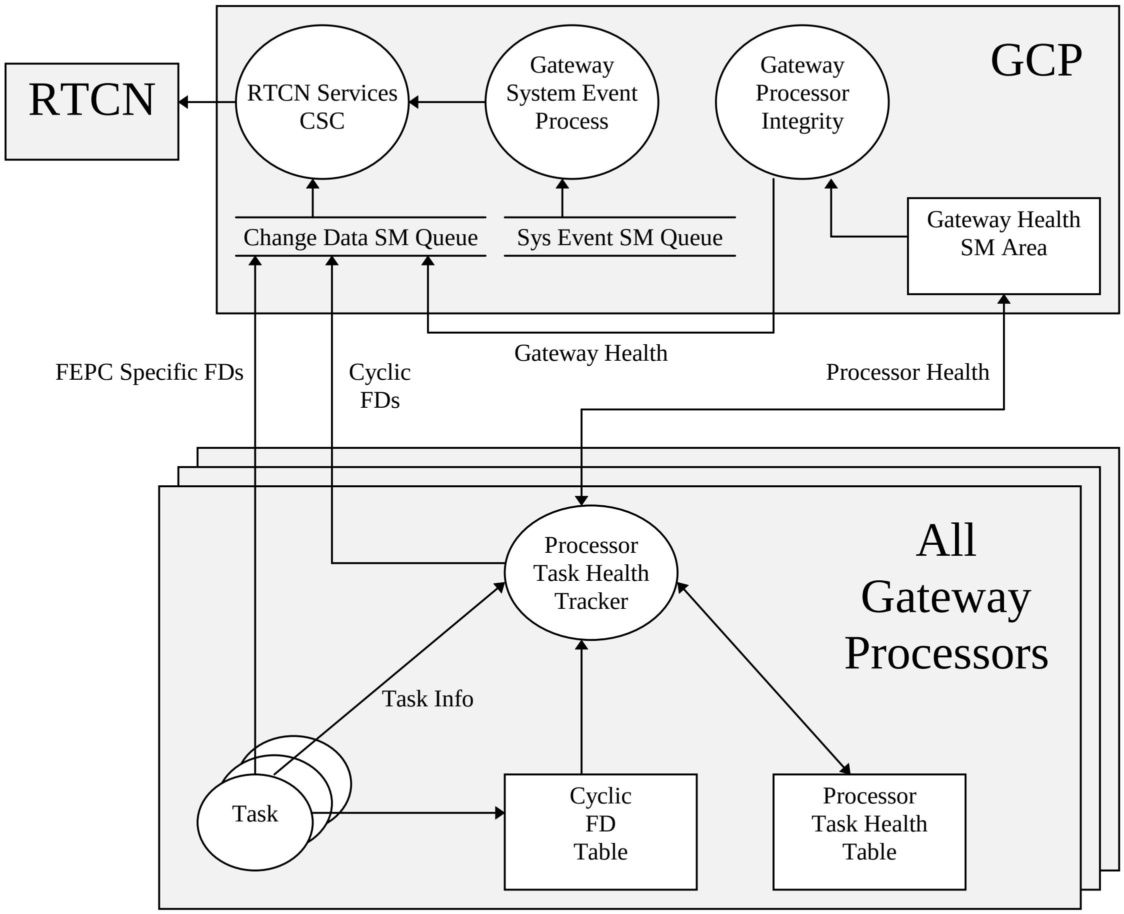
6.3 Gateway Subsystem Integrity CSC Design Specification

On each Gateway Processor, a task creation hook will be added during Gateway Initialization that will register all tasks with the local Processor Task Health Table as they are created. Also, a task deletion hook will be added during Gateway Initialization that will un-register tasks from the local Processor Task Health Table as they are deleted. By default, all tasks in the Processor Task Health Table are considered Critical. If a task is Non-critical, that task must explicitly call cgsTaskIsNoncritical() in its code. This will declare the task as Non-critical in the Processor Task Health Table. If a Critical task is found to be suspended or absent, Gateway Health and Status will discontinue processing, send a System Event code, send a System Message, and perform a Recovery. If a Non-critical task is found to be suspended or absent, Gateway Health and Status will send a System Message.

The Processor Task Health Tracker cycles through the Processor Task Health Table, retrieves the task information for each task, and stores the new information in the Table. The taskIdVerify() and taskIsSuspend()VxWorks function calls are used to verify the existence and processing state of each task. This data is updated in the Processor Task Health Table. If all tasks are functioning properly, the entire processor's health count is updated in the Health and Status Shared Memory Area located on the GCP. The Processor Task Health Tracker is also responsible for verifying the incrementing Health Count of the other processor boards. If a board fails to update its Health Count, the processor doing the check must take the appropriate action to initiate a switchover and recovery.

The Processor Health Tracker functions on each Gateway processor must also report their own task health in the form of a Health counter in the Gateway Health Common Shared Memory Area located on the Gateway Control Processor. The Gateway Processor Integrity task running on the GCP insures that all Processor Health tasks in the Gateway are healthy.

6.3.1 Gateway Subsystem Integrity CSC Detailed Data Flow



6.3.2 Gateway Subsystem Integrity CSC External Interfaces

6.3.2.1 Gateway Subsystem Integrity CSC Message Formats

6.3.2.1.1 Task is Absent

Message Number =
Message Group = CGS
Severity = Error

Gateway %s: Task %s with ID %d is absent.
This task is %s to the Gateway.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name
Insert #2 = Text string	VxWorks Task Name
Insert #3 = Integer	VxWorks Task Identifier
Insert #4 = Text string	Task Criticality (Critical, Non-critical)
Insert #5 = Integer	Gateway Error Databank Help ID

6.3.2.1.2 Task is Suspended

Message Number =
Message Group = CGS
Severity = Error

Gateway %s: Task %s with ID %d is suspended.
This task is %s to the Gateway.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name
Insert #2 = Text string	VxWorks Task Name
Insert #3 = Integer	VxWorks Task Identifier
Insert #4 = Text string	Task Criticality (Critical, Non-critical)
Insert #5 = Integer	Gateway Error Databank Help ID

6.3.2.1.3 Initiating Recovery

Message Number =
Message Group = CGS
Severity = Error

Gateway %s: Recovery has been initiated.
Reason Code: %d.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name
Insert #2 = Integer	Reason Code for a Recovery
Insert #3 = Integer	Gateway Error Databank Help ID

6.3.2.1.4 Processor Health Counters are not Updating

Message Number =
Message Group = CGS
Severity = Critical

Gateway %s: Processor Health counters are not updating.
Gateway Processor %d is not Healthy.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name
Insert #2 = Integer	Gateway Processor Number
Insert #3 = Integer	Gateway Error Databank Help ID

6.3.2.2 Gateway Subsystem Integrity CSC Display Formats

None

6.3.2.3 Gateway Subsystem Integrity CSC Input Formats

None

6.3.2.4 Gateway Subsystem Integrity CSC Recorded Data

None

6.3.2.5 Gateway Subsystem Integrity CSC Printer Formats

None

6.3.2.6 Gateway Subsystem Integrity CSC Interprocess Communications

6.3.2.6.1 System Event Codes (TBD)

6.3.2.7 Gateway Subsystem Integrity CSC External Interface Calls

6.3.2.7.1 Subsystem Failure and Recovery (as defined in Gateway Utility Services CSC)

6.3.2.7.2 System Event Codes (TBD)

6.3.2.7.3 Task Monitoring

Gateway FEPC Services includes a subsystem integrity task which will monitor the state of all tasks running on the FEPC. The operation of this task is transparent to the FEPC with one exception. The task monitor assumes all tasks on the FEPC are critical. If a critical task is detected to have terminated or suspended, the task monitor will declare a subsystem failure and perform recovery. If a task is deemed non-critical is should call the function described below to inform the task monitor function.

STATUS cgsTaskIsNoncritical ( );

Description	Informs the subsystem integrity task monitor that this task is not critical and should not cause a recovery if failed.
Parameters	None
Returns	OK or ERROR as defined in vxWorks.h

6.3.2.7.4 Periodic Measurement Generation

Gateway FEPC Services includes a subsystem integrity task which will output subsystem integrity FD's at a predetermined rate. A function is provided which will allow FEPC CSC's to register FD's for output. Once registered, the subsystem integrity task will check the FD for change at the predetermined rate and, if changed, send the new value to the GCP change data queue.

```
STATUS cgsRegisterCyclicFD (   char *fdNameSuffix,
                               void *fdAddr ,
                               UNSIGNED32 fdSize );
```

Description	Creates an entry in the Cyclic FD Table. Once in the Cyclic FD Table, the FD will be updated (if changed) at the Subsystem Integrity update rate.	
Parameters	fdNameSuffix	A pointer to the FD name suffix. If the FD name is "SG15SPPCA", the suffix is "SPPCA"

	fdAddr	Address of the FD to be output
	fdSize	Size of the FD in bytes
Returns	OK or ERROR as defined in vxWorks.h	

6.3.2.7.5 FDID Lookup

Some subsystem integrity FD's must be output immediately on change rather that at a rate. The CSC which owns an FD of this type is responsible for sending the data to the change data queue using the cgsWriteChangeData() call defined later in this document. A function is provided which will look up an FDID when given an FD name. This function should be called during subsystem initialization to look up all health/status FDs. The FDID may then be passed as a parameter to cgsWriteChangeData() or to the FEPC Measurement Processing function using TBD().

UNSIGNED32 cgsGetFdid(char *fdNameSuffix);

Description	Looks up an FDID when passed a FD name	
Parameters	fdNameSuffix	A pointer to the FD name suffix. If the FD name is “SG15SPPCA”, the suffix is “SPPCA”
Returns	FDID or NULL if not found	

6.3.2.7.6 Active / Standby Functions

All code running in the gateway should reference the same indicators to determine if the gateway is in active or standby mode. This startup mode is determined by the GCP from the SCT and placed in a shared memory variable. The following functions will access this variable and allow the FEPC to change it when a switchover command is received.

6.3.2.7.6.1 Test for Active Mode

int cgsIsActive();

Description	Test the Subsystem Integrity Mode variable for active mode
Parameters	None
Returns	TRUE if active mode, FALSE if not

6.3.2.7.6.2 Test for Standby Mode

int cgsIsStandby();

Description	Test the Subsystem Integrity Mode variable for standby mode
Parameters	None
Returns	TRUE if standby mode, FALSE if not

6.3.2.7.6.3 Test for Hot Spare Mode

int cgsIsHotSpare();

Description	Test the Subsystem Integrity Mode variable for hot spare mode
Parameters	None
Returns	TRUE if hot spare mode, FALSE if not

6.3.2.7.6.4 Test for Dual Mode (LDB mode)

int cgsIsDual();

Description	Test the Subsystem Integrity Mode variable for dual mode (LDB only)
Parameters	None
Returns	TRUE if dual mode, FALSE if not

6.3.2.7.6.5 Set the Subsystem Integrity Mode

STATUS cgsSetSsiMode(int mode);

Description	Set the Subsystem Integrity Mode variable to the specified mode	
Parameters	mode	0 = Active 1 = Standby
Returns	OK or ERROR as defined in vxWorks.h	

6.3.2.8 Gateway Subsystem Integrity CSC Table Formats

6.3.2.8.1 Processor Task Health Table

The Processor Task Health Table is a Table that is resident on every CPU in the Gateway. It tracks and maintains the processing state of each (non-VxWorks) task on the Gateway. Every update rate period, the Processor Task Health Table is updated with the current processing state of each task. Each entry in the Table contains the following parameters.

<Task ID> <Task Name> <Task Present> <Task State> <Critical>

Task ID:	The integer identifier VxWorks assigns to the task when it is created
Task Name:	The name given the task when it is created
Task Present:	A Boolean flag (True, False) value which states whether or not the task is present.
Task State:	The current VxWorks processing state of the task.
Critical:	A Boolean flag (True, False) which states whether or not the task is a Critical task.

6.3.2.8.2 Cyclic FD Table

The Cyclic FD Table is a Table that is resident on every CPU in the Gateway. FD entries are stored in this table when the registerCyclicFD() function call is made. The Cyclic FD Table stores the latest information on each FD entry in the Table and is updated every update rate period. Also every update rate period, the table entries are compared to the current FD values. If the values differ, the latest value is updated in the table and the change is reported through the Change Data shared memory message queue. Each entry in the Cyclic FD Table contains the following parameters:

<FD ID> <FD Name> <Address> <Size> <Latest Value>

FD ID:	The ID of the Function Designator entry.
FD Name:	The Name of the Function Designator entry.
Address:	The local Address of the Function Designator entry.
Size:	The Size of the Function Designator entry.
Latest Value:	The most recent value of the Function Designator entry.

6.3.3 Gateway Subsystem Integrity CSC Test Plan

6.3.3.1 Environment

The Common Gateway Services CIT will be performed in conjunction with the GSE Gateway Services CIT. A development Gateway will be configured as a GSE Gateway. RTCN commands will be sent to the development Gateway. The action taken and the responses returned will be verified.

6.3.3.2 Test Tools

The Gateway Integrated Test Environment CSC will be used to test the Gateway Subsystem Integrity CSC.

6.3.3.3 Test Cases

1. Verifying the transmission of Subsystem Integrity FDs every System Synchronous Rate period using the RTCN Change Data Analyzer.
2. Verify the Transmission and Reception of System Event codes.
3. Verify the initiation of recovery is a fatal error occurs.

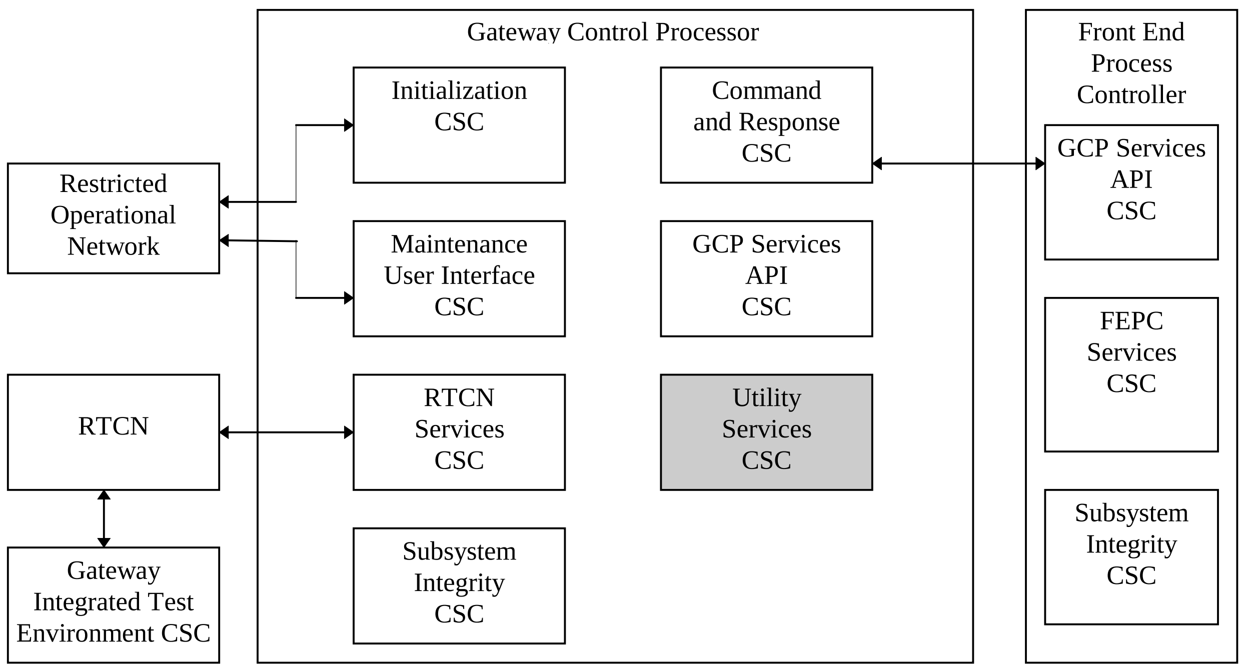
7. Gateway Utility Services CSC

7.1 Gateway Utility Services CSC Introduction

7.1.1 Gateway Utility Services CSC Overview

The Gateway Utility Services CSC is responsible for transmitting System Messages, error logging and status logging. It is part of the Common Gateway Services CSCI and is resident in the GCP.

Common Gateway Services CSCI



7.1.2 Gateway Utility Services CSC Operational Description

Gateway Utility Services CSC will provide the capability for any resource in the Gateway to generate System Messages via the RTCN. It will also provide the capability to log Error and Status messages to local disk. These error and status message can also be routed to the console port. Many metrics will be maintained by this CSC such as commands processed per second, etc. Gateway Utility Services also provides a Recovery Dump function which will dump all global and local variables to the disk.

7.2 Gateway Utility Services CSC Specifications

7.2.1 Gateway Utility Services CSC Groundrules

- Gateway Utility Services CSC will allow any resource in the Gateway to perform the below functions. These functions have been packaged with the GCP Services API CSC.
  - Generate System Messages,
  - Perform a Recovery Dump,
  - Log to local disk error/status messages,

- Route error/status messages to the console port.

- The message number used to format System Messages will be defined in a header file that will be provided by the System Message Services CSC.
- The Gateway Maintenance User Interface CSC shall provide the means to access the detailed Gateway error descriptions provided by the Gateway Utility Services CSC.
- The Gateway Maintenance User Interface CSC shall provide the means to decode the local version of Recovery Dump provided by the Gateway Utility Services CSC.

7.2.2 Gateway Utility Services CSC Functional Requirements

The Functional Requirements for the Gateway Utility Services CSC are arranged in the following major functions:

1. System Messages
2. Error/Status logging
3. Recovery Dump
4. External Interfaces

7.2.2.1 System Messages

The System Message routines provide all processes on the Gateway the ability to send a System Message.

1. Gateway Utility Services CSC shall provide an independent shared memory message queue and handling task for System Message processing.
2. Gateway Utility Services CSC shall provide a method for all resources on the Gateway to send System Messages.
3. Gateway Utility Services CSC shall provide the capability to specify System Message parameters.

7.2.2.2 Error/Status Logging

The Logging routines provide all processes on the Gateway the ability to log messages to the console port and/or the disk. Also, a means to register detailed descriptions of errors will be provided.

1. Gateway Utility Services CSC shall provide an independent shared memory message queue and handling task for Log Message processing.
2. Gateway Utility Services CSC shall provide the means to log a message to SDC.
3. Gateway Utility Services CSC shall provide a method for all resources on the Gateway to Log Error/Status messages to the Gateway disk and/or the console port.
4. Gateway Utility Services CSC shall provide the capability to specify Error/Status message parameters.
5. Gateway Utility Services shall provide a means to register a detailed description of error messages during development.
6. Detailed descriptions of error messages shall be stored as a database on the Gateway disk and will be accessible via the Gateway Maintenance User Interface CSC.
7. Gateway Utility Services CSC shall provide a header file mapping Gateway Errors to the Error/Status codes which have detailed descriptions in the database.
8. If an error has a detailed description associated with it, this shall be made known using a system message which references the database entry by message number.

7.2.2.3 Recovery Dump

A Recovery Dump occurs when a fatal error has been encountered on the Gateway. It is a means of dumping raw Gateway information to the Gateway disk before the Gateway is terminated.

1. Gateway Utility Services CSC shall perform a Recovery Dump whenever a fatal error occurs in the Gateway.
2. Recovery Dumps shall be sent to SDC first, and then logged locally.
3. When a Recovery Dump occurs, all Gateway resident processors shall write the following information to the SDC and to the disk:
   3.1. All variables (global and local).
   3.2. Route Table, Transaction Table, and Streams Table (if GCP).
   3.3. All loaded TCID Tables (if FEPC).
   3.4. Reason for failure (if known).

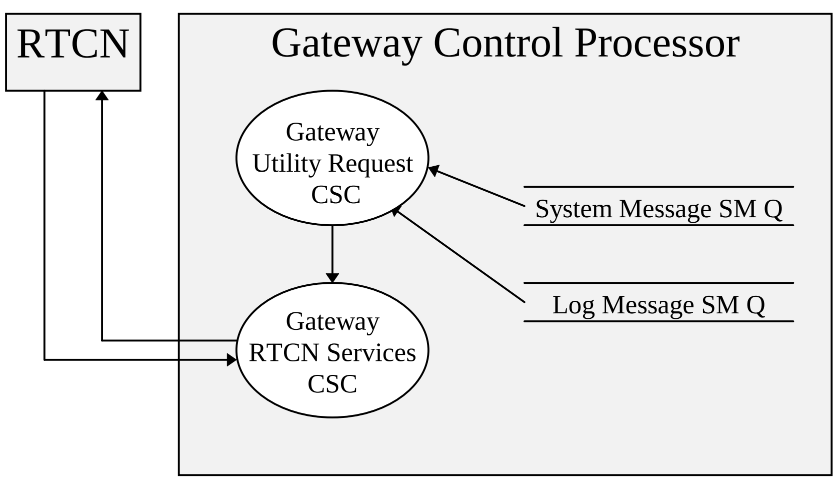
7.2.2.4 External Interfaces

1. Gateway Utility Services CSC shall track the state and performance of all network interfaces.
2. Gateway Utility Services CSC shall track the state and performance of the Gateway Disk.
3. Gateway Utility Services CSC shall be responsible for monitoring the following for each network interface:
   3.1. Number of packets transmitted each second
   3.2. Highest number of packets transmitted in a second
   3.3. Number of bytes transmitted each second
   3.4. Highest number of bytes transmitted in a second
   3.5. Number of packets received each second
   3.6. Highest number of packets received in a second
   3.7. Number of bytes received each second
   3.8. Highest number of bytes received in a second
   3.9. Number of errors
4. Gateway Utility Services CSC shall be responsible for monitoring the following for the Gateway Disk:
   4.1. Use counter
   4.2. Error counter

7.2.3 Gateway Utility Services CSC Performance Requirements

No performance requirements have been identified for the Gateway Utility Services CSC for the Thor delivery.

7.2.4 Gateway Utility Services CSC Interfaces Data Flow Diagram



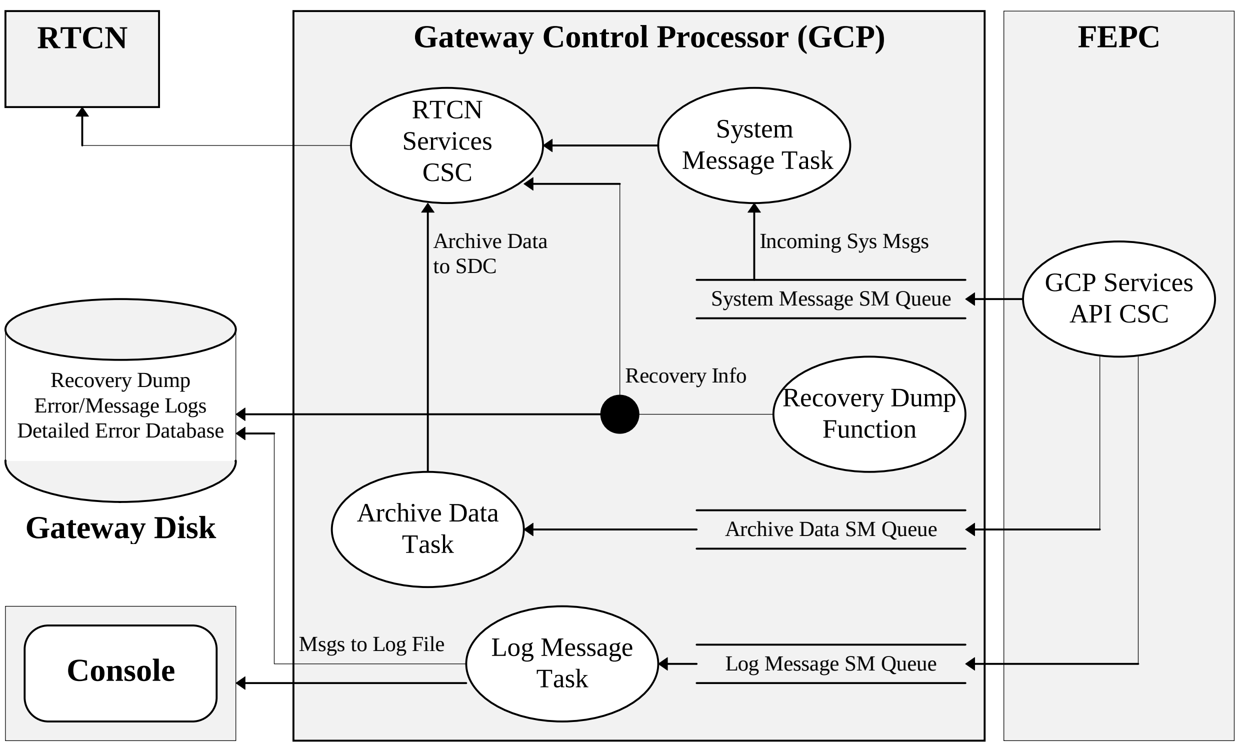
The Utility Services CSC is spawned by the Gateway Initialization CSC. The System Message, and Log Message Shared Memory Message Queues provide every resource in the Gateway with access to those services. All Utility Services CSC services that require the RTCN are routed to the Gateway RTCN Services CSC.

7.3 Gateway Utility Services CSC Design Specification

The Gateway Utility Services CSC tasks monitor the Log Message and System Message Shared Memory Message Queues. When entries are found in either queue, the Gateway Utility Services CSC reads them and forwards them to the appropriate resource. Logged messages may be routed to any combination of the Console port, SDC, or the Gateway Disk. All System Messages are sent over the RTCN. The Recovery Dump function outputs all Recovery Data first to SDC and then to the local Gateway Disk.

The detailed message database is stored on the local Gateway Disk. If a detailed message is requested, the disk is accessed and the message is retrieved. This database is generated during development. Developers access the Gateway Error Catalog WWW site and submit the necessary information to create a detailed error message entry. A header file is created which associates that error message with a numerical index and a mnemonic. These are used later to index the database.

7.3.1 Gateway Utility Services CSC Detailed Data Flow



7.3.2 Gateway Utility Services CSC External Interfaces

7.3.2.1 Gateway Utility Services CSC Message Formats

7.3.2.1.1 Log Message Error

Message Number =
Message Group = CGS
Severity = Warning

Gateway %s: Error logging message to the Gateway Disk.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name
Insert #3 = Integer	Gateway Error Databank Help ID

7.3.2.1.2 Network Interface Inaccessible

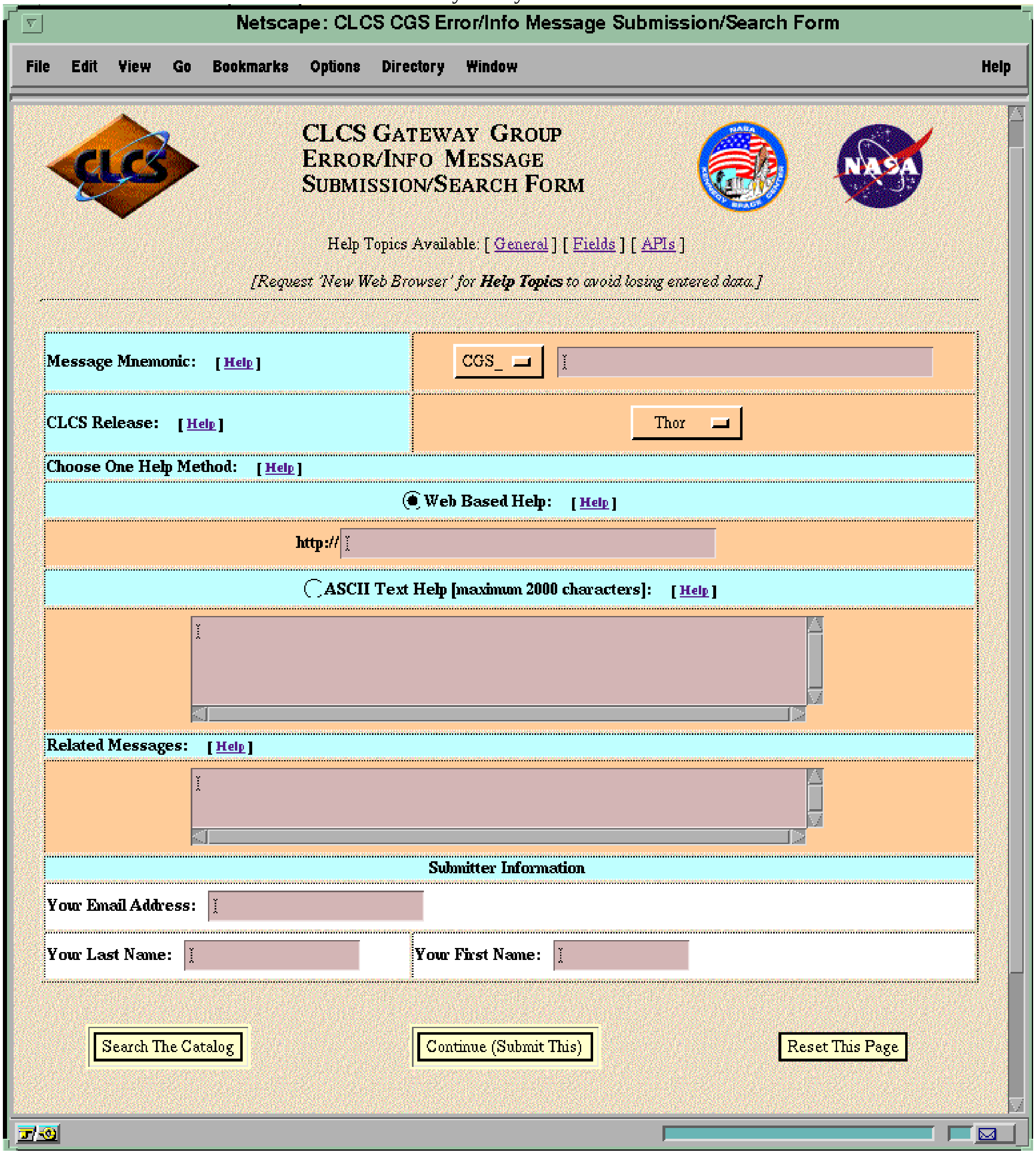
Message Number =
Message Group = CGS
Severity = Error

Gateway %s: Network Interface %s is inaccessible.
Gateway Error: %d.

Insert #1 = Text string	Gateway Logical Name
Insert #2 = Text string	VxWorks interface name
Insert #3 = Integer	Gateway Error Databank Help ID

7.3.2.2 Gateway Utility Services CSC Display Formats

The following is a sample display format for the Gateway team’s detailed error database.



7.3.2.3 Gateway Utility Services CSC Input Formats

None

7.3.2.4 Gateway Utility Services CSC Recorded Data

7.3.2.4.1 Recovery Dump

The Recovery Dump is a snapshot of a processor's memory at the time when a fatal error occurred. The format for this is not yet defined.

7.3.2.5 Gateway Utility Services CSC Printer Formats

None

7.3.2.6 Gateway Utility Services CSC Interprocess Communications

7.3.2.7 Gateway Utility Services CSC External Interface Calls

7.3.2.7.1 Log Message (as defined in the Gateway GCP Services API CSC)

7.3.2.7.2 System Message (as defined in the Gateway GCP Services API CSC)

7.3.2.8 Gateway Utility Services CSC Table Formats

None

7.3.3 Gateway Utility Services CSC Test Plan

7.3.3.1 Environment

The Common Gateway Services CIT will be performed in conjunction with the GSE Gateway Services CIT. A development Gateway will be configured as a GSE Gateway. RTCN commands will be sent to the development Gateway. The action taken and the responses returned will be verified.

7.3.3.2 Test Tools

The Gateway Integrated Test Environment will be used to test the Utility Services CSC.

7.3.3.3 Test Cases

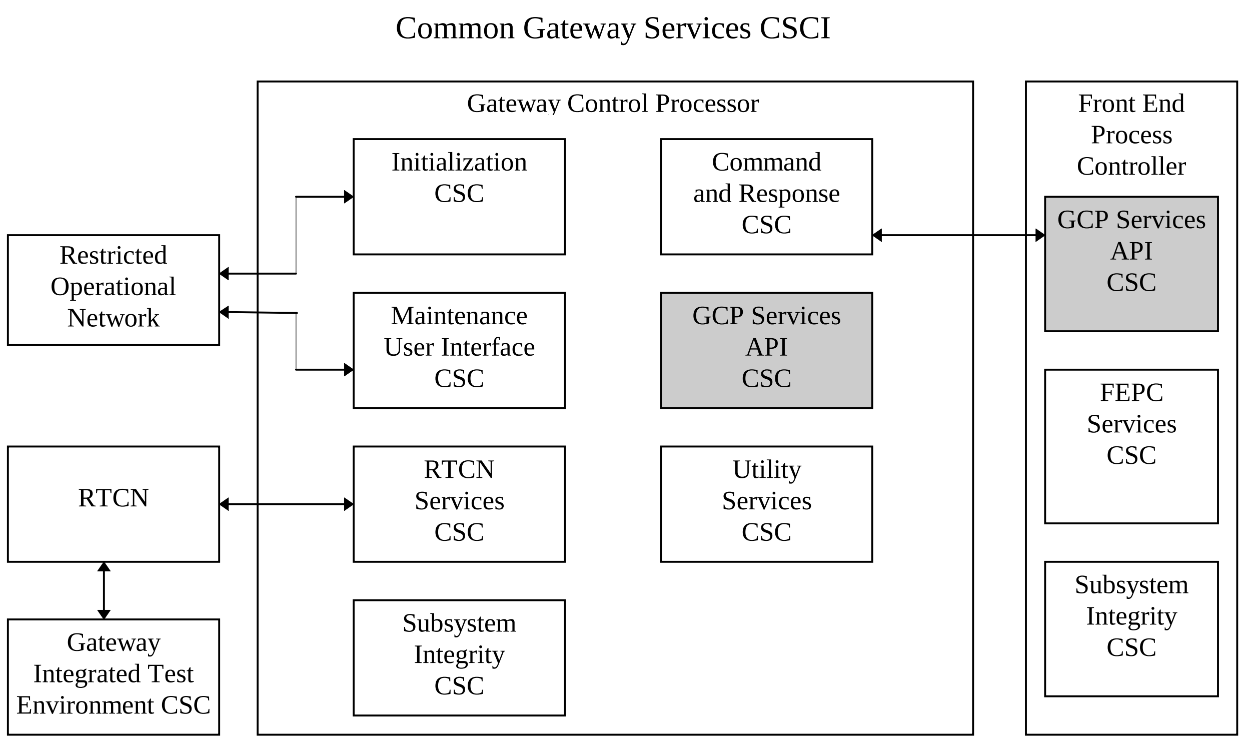
1. Verify the existence and update of the log file.
2. Using the RTCN Analyzer, verify the transmission of System Messages and Archive Data.
3. Perform a Recovery Dump from the Console and verify local and remote formats.
4. Manually disable a network interface and verify that this CSC tracks it.

8. Gateway GCP Services API CSC

8.1 Gateway GCP Services API CSC Introduction

8.1.1 Gateway GCP Services API CSC Overview

The Gateway GCP Services API CSC is responsible for providing a common interface from the Front End Process Controller (FEPC) to the Gateway Control Processor (GCP) and is resident on the FEPC.



8.1.2 Gateway GCP Services API CSC Operational Description

This CSC provides a common interface to services provided by the GCP. These services include receiving commands from the RTCN and generating their response, sending change data or system messages, logging messages to the local hard drive or display, etc.

8.2 Gateway GCP Services API CSC Specifications

8.2.1 Gateway GCP Services API CSC Groundrules

- None

8.2.2 Gateway GCP Services API CSC Functional Requirements

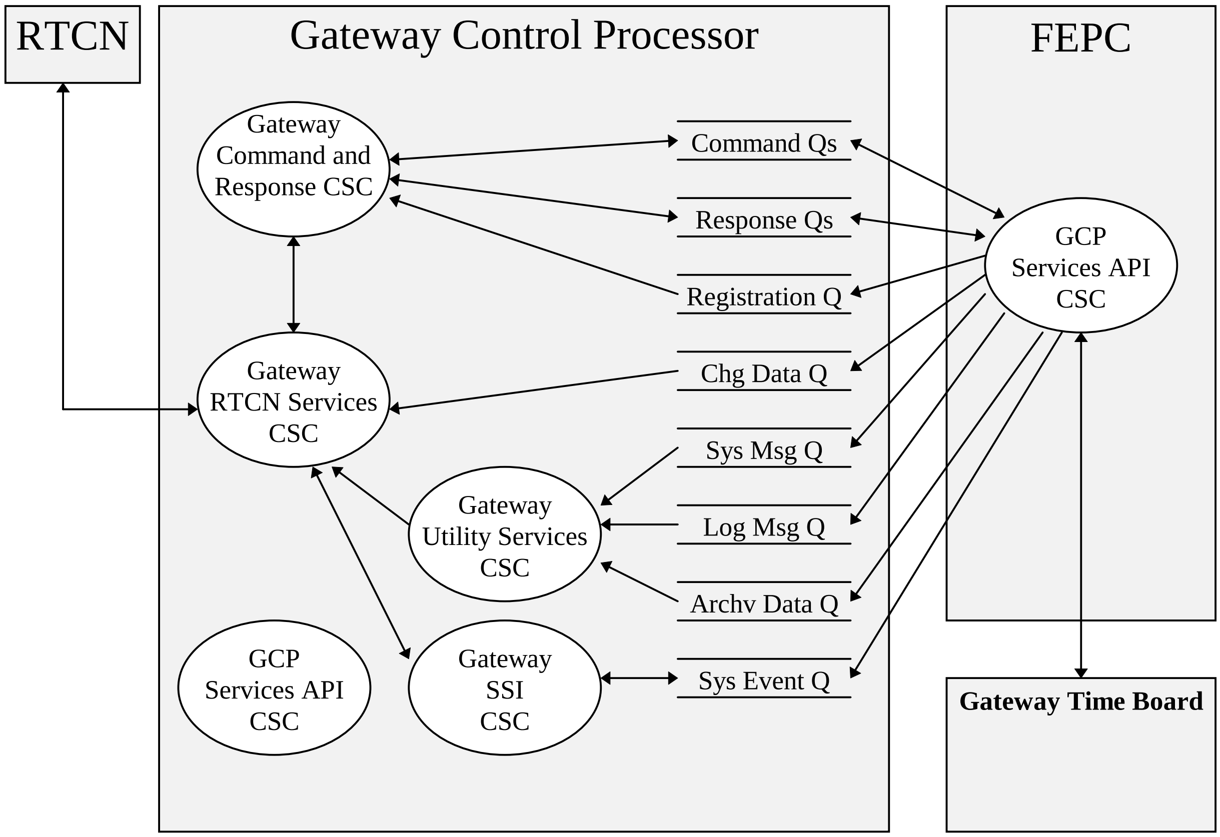
1. The GCP Services API will provide an interface for receiving commands from the RTCN.
2. The GCP Services API will provide a normal and high priority queue for receiving commands

3. The GCP Services API will provide an interface for sending commands to the RTCN.
4. The GCP Services API will provide an interface for receiving responses from the RTCN
5. The GCP Services API will provide an interface for generating command responses to the RTCN
6. The GCP Services API will provide an interface for sending measurement change data to the RTCN
7. The GCP Services API will provide an interface for sending system messages to the RTCN
8. The GCP Services API will provide an interface for sending system event codes to the RTCN
9. The GCP Services API will provide an interface for sending archive data to the RTCN
10. The GCP Services API will provide an interface for sending log messages to the local hard drive and/or to the console port and to the RTCN.
11. The GCP Services API will provide an interface for reading time.
12. Gateway GCP Services API CSC shall provide a function to access the following:
    - 12.1. Time in BCD with microsecond resolution
    - 12.2. Millisecond Time of Day (32-bit integer)
    - 12.3. Julian Time of Year (32-bit integer)

~~8.2.3~~ Gateway GCP Services API CSC Performance Requirements

1. Gateway GCP Services API CSC will return time within 10 μ seconds from the time of a request.

8.2.4 Gateway GCP Services API CSC Interfaces Data Flow Diagram



The GCP Services API CSC is initialized by the Gateway Initialization CSC. After Initialization, any Gateway resident processor (including the GCP) may use the API by registering through the Registration Shared Memory Message Queue. After registering, the capability to perform Command and Response functions may be accomplished through these Queues. They provide a direct link to the Gateway Command and Response, and Gateway RTCN Services CSCs.

8.3 Gateway GCP Services API CSC Design Specification

The Gateway GCP Services API CSC is the programming interface used for communicating with the GCP. It is the interface that allows all resources in the Gateway access to Common Gateway functions. The API communicates with the GCP using shared memory message queues, semaphores, and shared memory areas. All Command/Response, Change Data generation, and Utility Services seen by the FEPC are accomplished using this API. It a set of functions for accessing and transferring to these shared memory devices. The Gateway GCP Services API is also responsible for providing Time board reading functions.

8.3.1 Gateway GCP Services API CSC Detailed Data Flow

See the Interfaces Data Flow Diagram.

8.3.2 Gateway GCP Services API CSC External Interfaces

8.3.2.1 Gateway GCP Services API CSC Message Formats

None

8.3.2.2 Gateway GCP Services API CSC Display Formats

None

8.3.2.3 Gateway GCP Services API CSC Input Formats

None

8.3.2.4 Gateway GCP Services API CSC Recorded Data

None

8.3.2.5 Gateway GCP Services API CSC Printer Formats

None

8.3.2.6 Gateway GCP Services API CSC Interprocess Communications

None

8.3.2.7 Gateway GCP Services API CSC External Interface Calls

8.3.2.7.1 GCP Services Initialization

STATUS cgsGcpServicesInit();

Description: Performs all initialization required by GCP Services. This function is called by the initialization service of Gateway FEPC Services

Parameters: None

Returns: OK or ERROR as defined in vxWorks.h

8.3.2.7.2 Command/Response Interface

This interface provides the ability for the FEPC to receive or send commands from or to the RTCN.

8.3.2.7.2.1 The CGS_COMMAND_INFO_TYPE structure

The CGS_COMMAND_INFO_TYPE structure is used by all function calls in the command/response interface. The structure contains selected fields from the RTCN header.

```
typedef struct
{
        UNSIGNED16 logicalSource;
        UNSIGNED16 routeCode;
        UNSIGNED16 requestId;
        UNSIGNED16 transactionId;
        UNSIGNED16 responsibleSystem;
        UNSIGNED16 logicalDestination
        INTEGER16    completionCode;
        UNSIGNED16 byteCount;
        UNSIGNED32 mstod;
        UNSIGNED32 issueTimeout;
        UNSIGNED32 transIndex;
} CGS_COMMAND_INFO_TYPE;
```

logicalSource	The CPU ID of the processor which sent this command. These ID's are defined in TBD header file.
routeCode	Identifies the destination process for a specific command. One process may respond to multiple route codes. Current plan is to use the route codes defined by CCMS. The process must respond to all request ID's within each route code. If a request ID is not required to be processed by this CPU, a successful response should be generated.
requestId	Identifies the command to be executed by the process under this request id. Current plan is to use the request id's defined by CCMS.
transactionId	The CCP transaction ID
responsibleSystem	The system ID for the system which issued this command
logicalDestination	The CPU ID of the destination for this command. These ID's are defined in TBD header file.
completionCode	Completion status for this command. This parameter must be filled in prior to performing the csgGenerateResponse() call.
byteCount	Size of the command payload body in bytes. This parameter must be modified to reflect the size of the response payload body prior to performing the csgGenerateResponse() call.
mstod	Millisecond time of day. Time command was received.
issueTimeout	If a command requires a significant amount of time to process, an issued response should be generated immediately followed by a completion (success or fail) response when the command completes. This field should be modified to be the time in milliseconds that the command will require to complete prior to sending the issued response. This field is not used by all other responses.
transIndex	Used by the GCP to track command responses. This field must not be modified.

8.3.2.7.2.2 Copy Header

```
void cgsCopyInfo(        CGS_COMMAND_INFO_TYPE * from,
```

CGS_COMMAND_INFO_TYPE *to);

Description: Copies data from one CGS_COMMAND_INFO_TYPE structure to another.

Parameters: from, to — pointers to CGS_COMMAND_INFO_TYPE structures. The <from> structure will be copied into the <to> structure.

Returns: none

8.3.2.7.2.3 Wait Command

STATUS cgsWaitCommand(CGS_COMMAND_INFO_TYPE *info,
void *buffer);

Description: This routine will pend waiting for a command from the RTCN. The command information and data is returned to the caller.

Parameters: header — pointer to a CGS_COMMAND_INFO_TYPE structure which is filled in as part of the call.

buffer — pointer to a buffer to be filled in with the command payload body

Returns: OK or ERROR as defined in vxWorks.h

8.3.2.7.2.4 Generate Response

STATUS cgsGenerateResponse(CGS_COMMAND_INFO_TYPE *info,
void *buffer);

Description: Generates a response message to the subsystem which originated the command contained in the command header structure. This function will also handle generating the time tag for the response.

Parameters: header — pointer to the CGS_COMMAND_INFO_TYPE structure which was returned from the cgsWaitCommand()call. The sizeInBytes parameter in this structure must be modified to reflect the size of the response data. The completionCode parameter must be updated with the proper completion status. If the completion code is an issued response, the issueTimeout parameter must be set to the time in milliseconds the command will require to complete. The remainder of the structure must be left unmodified.

buffer — pointer to a buffer which contains the response data.

Returns: OK or ERROR as defined in vxWorks.h

8.3.2.7.2.5 Generate Command

STATUS cgsGenerateCommand(CGS_COMMAND_INFO_TYPE *info,
void *buffer, int destination, int flags, int priority);

Description: Sends a command to the subsystem specified by the destination parameter.

Parameters:	header	pointer to a CGS_COMMAND_INFO_TYPE structure. The following parameters must be filled in prior to the call: routeCode, requestId, sizeInBytes
	destination	logical CPU ID for the destination. Defined in ** TBD **
	flags	message options flags. Defined in ** TBD **
	buffer	pointer to a buffer which contains the command data.
	priority	CGS_NORMAL_PRIORITY or CGS_HIGH_PRIORITY

Returns: OK or ERROR as defined in vxWorks.h

8.3.2.7.2.6 Get Response

```
STATUS cgsGetResponse(    CGS_COMMAND_INFO_TYPE *info,
                          void *buffer, int timeout);
```

Description: Reads a command response from the response queue.

Parameters:	header	pointer to a CGS_COMMAND_INFO_TYPE structure which is filled in as part of the call
	buffer	pointer to a buffer to be filled in with the response data
	timeout	time in system clock ticks to wait for the response

Returns: OK or ERROR as defined in vxWorks.h

8.3.2.7.3 Change Data Interface

The change data interface provides the mechanism for sending change data to the GCP where it will be formatted and placed in a change data packet for transmission to the remainder of the system.

8.3.2.7.3.1 Measurement Status

Measurement status is currently an integer with two bits defined as follows:

FD processing fail / no fail
FD warning / no warning

These bits are defined in the header file as:
CGS_STATUS_FD_FAIL
CGS_STATUS_FD_NO_FAIL
CGS_STATUS_FD_WARNING
CGS_STATUS_FD_NO_WARNING

8.3.2.7.3.2 Write Change Data

```
STATUS cgsWriteChangeData(    UNSIGNED32 status,
                              UNSIGNED32 fdid,
                              UNSIGNED32 sizeInBytes,
                              void *data);
```

Description:	Outputs change data for placement in the change data packet.	
Parameters:	status	Measurement status flags as defined above.
	fdid	Function Designator ID from the command or measurement data table.
	sizeInBytes	size of the data element in bytes
	data	pointer to data element
Returns:	OK or ERROR as defined in vxWorks.h	

8.3.2.7.4 Logging Interface

The message logging interface allows the FEPC to pass messages to the GCP for output to the console port, logging to a local disk file or output on the RTCN as a system message. Two versions are provided. The first allows application provided text to be output to the console port and/or disk file. The second outputs a system message and utilized a message number from the system message catalog. The message number passed to this routine should be defined in a header file that is used by the GCP and by the CLCS CCP to format the text for output.

8.3.2.7.4.1 Log Message

```
STATUS cgsLogMessage(int msgType,
                     char *format,
                     [,arg]…);
```

Description:	Passes a text message to the GCP which be logged to the console port and/or the local disk.	
Parameters:	msgType	Identifies the message type. A bit pattern defined in the header file as: CGS_DISPLAY_MSG_TYPE CGS_DISK_MSG_TYPE or both via: CGS_LOCAL_MSG_TYPE
	format	printf compatible text format string
	[arg]…	printf compatible argument list
Returns:	OK or ERROR as defined in vxWorks.h	

8.3.2.7.4.2 Archive Data Logging (BFL)

```
STATUS cgsArchiveData(void *data,
                      int byteCount);
```

Description:	Logs the specified data to SDC for archival purposes	
Parameters:	data	Pointer to the data to be logged to SDC.
	byteCount	Size of the data to be logged in bytes.
Returns:	OK or ERROR as defined in vxWorks.h	

8.3.2.7.4.3 System Message

```
STATUS cgsSystemMessage (     int msgNumber,
                              int numInserts,
                              [,arg]… );
```

Description: Passes message identifier information to the GCP which will format and output a system message.

Parameters:

	msgNumber	Message number as defined in the system message catalog.
	numInserts	Number of inserts for this message. Must agree with the number defined in the system message catalog
	[arg]…	Two parameters for each insert. The first is the insert type as defined below. The second is the insert.

Returns: OK or ERROR as defined in vxWorks.h

Insert types: The following insert types are defined:

ASCIIZ_INSERT	ASCII text string
INTEGER_INSERT	32-bit integer
FLOAT_INSERT	32-bit floating point
CDT_INSERT	Count down time
GMT_INSERT	UTC
INTEGER64_INSERT	64-bit integer
FLOAT64_INSERT	64-bit floating point

Example:

```
cgsSystemMessage(     GSE_HIM_RESPONSE_ERROR, 4,
                      ASCIIZ_INSERT,         “Manchester”,
                      INTEGER_INSERT,        (int) him_address,
                      INTEGER_INSERT,        (int) him_channel,
                      INTEGER_INSERT,        (int) him_data);
```

8.3.2.7.5 UTC Interface

The UTC interface provides a mechanism to read UTC.

8.3.2.7.5.1 CGS_TIME_TYPE structure

The cgsGetTime() function fills in a CGS_TIME_TYPE structure. This structure is defined as follows:

```
typedef struct
{
        UNSIGNED16 usec100;
        UNSIGNED16 status;
        UNSIGNED16 jtoy;
        UNSIGNED16 year
        UNSIGNED32 mstod;
} CGS_ TIME_TYPE;
```

8.3.2.7.5.2 Get Time

STATUS cgsGetTime(CGS_TIME_TYPE *time);

Description: Reads the time card and returns current time in a CGS_TIME_TYPE structure.

Parameters: time a pointer to a CGS_TIME_TYPE structure which will be filled in by the call.

Returns: OK or ERROR as defined in vxWorks.h

8.3.2.7.5.3 Get MSTOD

UNSIGNED32 cgsGetMstod(CGS_TIME_TYPE *time);

Description: Reads the time card and returns current time in a CGS_TIME_TYPE structure. Is faster than cgsGetTime() since jtoy and year are not returned

Parameters: time a pointer to a CGS_TIME_TYPE structure which will be filled in by the call. This call will fill in only the mstod, usec100 and status parameters. May be specified as NULL if mstod is all that is required.

Returns: mstod Current millisecond time-of-day

8.3.2.7.5.4 Get ASCII Time

char *cgsGetAsciiTime(char *returnString, char *formatString)

Description: Reads current time and generates an ascii character representation of time based on a caller provided format string

Parameters: returnString Pointer to a character string into which the formatted time text will be placed

formatString Pointer to a character string containing format codes (defined below)

Returns Pointer to returnString

Valid format codes are:

%n	insert a new-line character
%t	insert a tab character
%m	month of year - 01 to 12
%d	day of month - 01 to 31
%y	2 digit year - 00 to 99
%Y	4 digit year - 0000 to 9999
%D	date as mm/dd/yy
%H	hour - 00 to 23
%M	minute - 00 to 59
%S	second - 00 to 59
%T	24 hour time as HH:MM:SS.mmm
%j	day of year - 001 to 366
%w	day of week - Sunday = 0
%a	abbreviated weekday - Sun to Sat

%h	abbreviated month - Jan to Dec
%r	12 hour time as HH:MM:SS.mmm XM
%u	microseconds - 0 to 999

Examples: Format string is "%h. %d, %Y"
Return string is "Feb. 04, 1997"

Format string is "%j/%T"
Return string is "035/15:26:32.059"

8.3.2.7.5.5 ASCII Time

```
char *cgsAsciiTime(     CGS_TIME_TYPE *time,
                        char *returnString,
                        char *formatString)
```

Description: same as cgsGetAsciiTTime() except this call uses a CGS_TIME_TYPE structure that has already been filled in by a previous cgsGetTime() call.

8.3.2.7.6 Health and Status Interface

This interface identifies shared memory areas which will contain health and status information. In addition, this interface provides functions which may be used to look up the FDID of a subsystem health and status measurement.

The shared memory area contains two partitions; a common region and a gateway specific subsystem health and status region. Each gateway type (LDB, GSE, PCM, etc.) will define a unique subsystem health and status region This region will be available via HTTP over the network

8.3.2.7.6.1 Get Pointer to Subsystem Status Area

void * cgsGetHsPointer ();

Description: Gets a pointer to the gateway specific subsystem health and status area.

Parameters: none

Returns: pointer to the memory area or NULL if the shared memory area was not found.

8.3.2.8 Gateway GCP Services API CSC Table Formats

None

8.3.3 Gateway GCP Services API CSC Test Plan

8.3.3.1 Environment

The Common Gateway Services CIT will be performed in conjunction with the GSE Gateway Services CIT. A development Gateway will be configured as a GSE Gateway.

8.3.3.2 Test Tools

The GCP Simulator software will be used to test the GCP Services API CSC.

8.3.3.3 Test Cases

1. Sending a command to the FEPC and receiving a response.
2. Reading out Change Data from the appropriate queue.

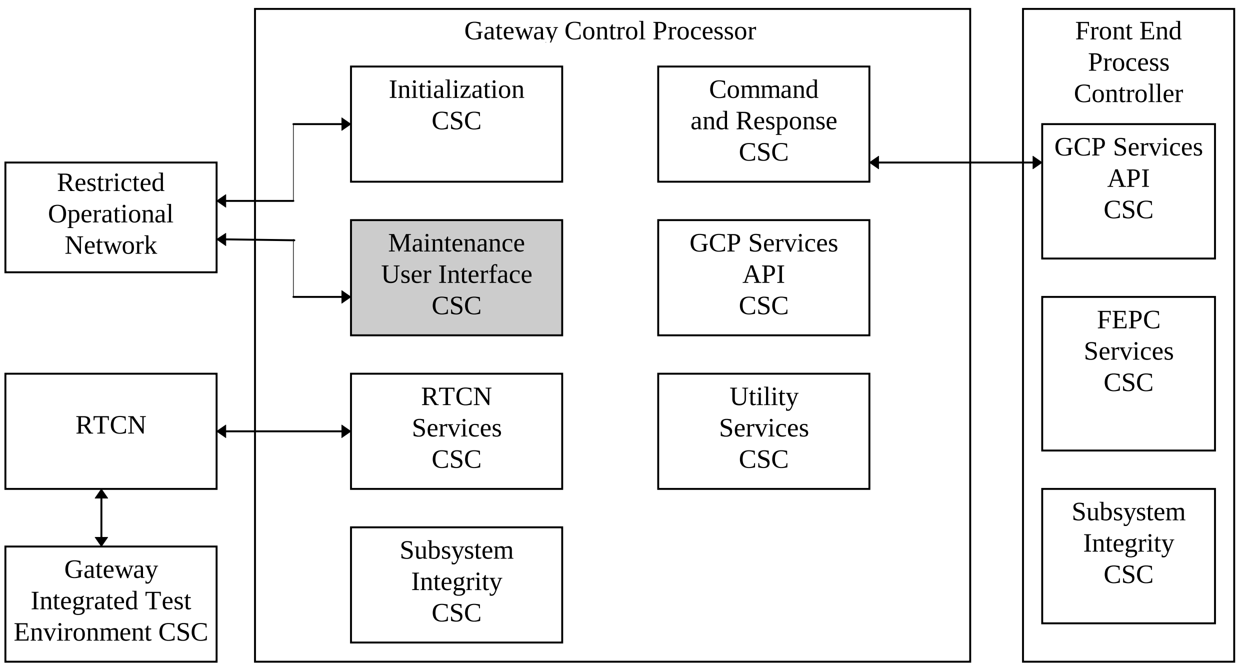
9. Gateway Maintenance User Interface CSC

9.1 Gateway Maintenance User Interface CSC Introduction

9.1.1 Gateway Maintenance User Interface CSC Overview

The Gateway Maintenance User Interface CSC allows access to the Gateway via the Restricted Operational Network. It is part of the Common Gateway Services CSCI and is resident in the GCP.

Common Gateway Services CSCI



9.1.2 Gateway Maintenance User Interface CSC Operational Description

Gateway Maintenance User Interface CSC will allow access to the Gateway via the Restricted Operational Network in order to access several Gateway maintenance features. It will allow access to Network statistics, Command and Data processing statistics, static tables resident on the Gateway, verbose error descriptions, and Health and Status information. The Interface will also have the capability to call some Gateway commands.

9.2 Gateway Maintenance User Interface CSC Specifications

9.2.1 Gateway Maintenance User Interface CSC Groundrules

- The Gateway Maintenance User Interface CSC will not be available in Primitive mode.
- Updates to the Gateway Maintenance User Interface CSC's counts and table displays will be no more frequent than once per second.
- The Gateway Maintenance User Interface CSC will use the internal Route Code 0 to send commands to the FEPC in the Gateway.

- The Gateway Maintenance User Interface CSC will be available only over the Restricted Operational Network.
- The GITE will not be used in an operational environment.

9.2.2 Gateway Maintenance User Interface CSC Functional Requirements

The Functional Requirements for the Gateway Maintenance User Interface CSC are arranged in the following major functions:

1. Initialization
2. Command and Response
3. RTCN Services
4. Timer Services
5. Subsystem Integrity
6. Utility Services
7. FEPC Specific

9.2.2.1 Initialization

The Initialization aspect of the Gateway Maintenance User Interface CSC is the set of the CSC's capabilities that pertains to the Gateway Initialization CSC.

1. The Gateway Maintenance User Interface CSC shall be able to display Gateway Control Processor (GCP) and Front End Processor Controller (FEPC) Initialization modes.
2. The Gateway Maintenance User Interface CSC shall be able to display the TCIDs available on the Gateway disk.
3. The Gateway Maintenance User Interface CSC shall be able to display ASCII Tables located on the Gateway disk (e.g., TCID Tables, Multicast Streams Table).

9.2.2.2 Command and Response

The Command and Response aspect of the Gateway Maintenance User Interface CSC is the set of the CSC's capabilities that pertains to the Gateway Command and Response CSC.

1. The Gateway Maintenance User Interface CSC shall be able to send internal commands to the FEPC using the Gateway Command and Response CSC.
2. The Gateway Maintenance User Interface CSC shall be able to track display command and response performance statistics.

9.2.2.3 RTCN Services

The RTCN Services aspect of the Gateway Maintenance User Interface CSC is the set of the CSC's capabilities that pertains to the Gateway RTCN Services CSC.

1. The Gateway Maintenance User Interface CSC shall be able to display the Network Streams Table provided by the Gateway RTCN Services CSC.
2. The Gateway Maintenance User Interface CSC shall be able to display RTCN Network statistics for the Gateway.

9.2.2.4 Timer Services

The Timer Services aspect of the Gateway Maintenance User Interface CSC is the set of the CSC's capabilities that pertains to the Gateway Timer Services CSC.

1. The Gateway Maintenance User Interface CSC shall be capable of displaying the current Gateway time.
2. The Gateway Maintenance User Interface CSC shall be capable of setting the Gateway's Time Of Day clock.

9.2.2.5 Subsystem Integrity

The Subsystem Integrity aspect of the Gateway Maintenance User Interface CSC is the set of the CSC's capabilities that pertains to the Gateway Subsystem Integrity CSC.

1. The Gateway Maintenance User Interface CSC shall have read access to all board health counts.

9.2.2.6 Utility Services

The Utility Services aspect of the Gateway Maintenance User Interface CSC is the set of the CSC's capabilities that pertains to the Gateway Utility Services CSC.

1. The Gateway Maintenance User Interface CSC shall be able to display System Message, System Event, and Error counts.
2. The Gateway Maintenance User Interface CSC shall be able to display Restricted Operational Network statistics for the Gateway.
3. The Gateway Maintenance User Interface CSC shall be able to display Shared Memory Message Queue information.
4. The Gateway Maintenance User Interface CSC shall be capable of decoding the local version of a Recovery Dump provided by Gateway Utility Services CSC.
5. The Gateway Maintenance User Interface CSC shall be capable of displaying the detailed error descriptions provided by the Gateway Utility Services CSC.

9.2.2.7 FEPC Specific

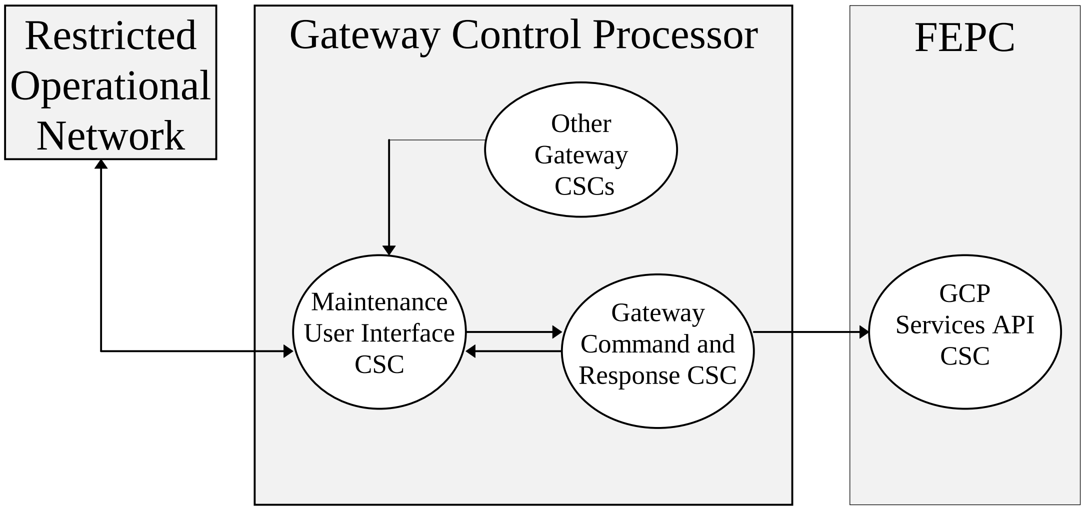
The FEPC Specific aspect of the Gateway Maintenance User Interface CSC is the set of the CSC's capabilities that are unique to the type of FEPC in the Gateway.

1. The Gateway Maintenance User Interface CSC shall be able to decode and display information from the Gateway specific Shared Memory Area.

9.2.3 Gateway Maintenance User Interface CSC Performance Requirements

No performance requirements have been identified for the Gateway Maintenance User Interface CSC for the Thor delivery.

9.2.4 Gateway Maintenance User Interface CSC Interfaces Data Flow Diagram



The Gateway Maintenance User Interface CSC is a server which handles client requests over the Restricted Operational Network. Requests which require commands to the FEPC are implemented using the Gateway Command and Response CSC. All other requests are handled by reading information from the other Gateway CSCs.

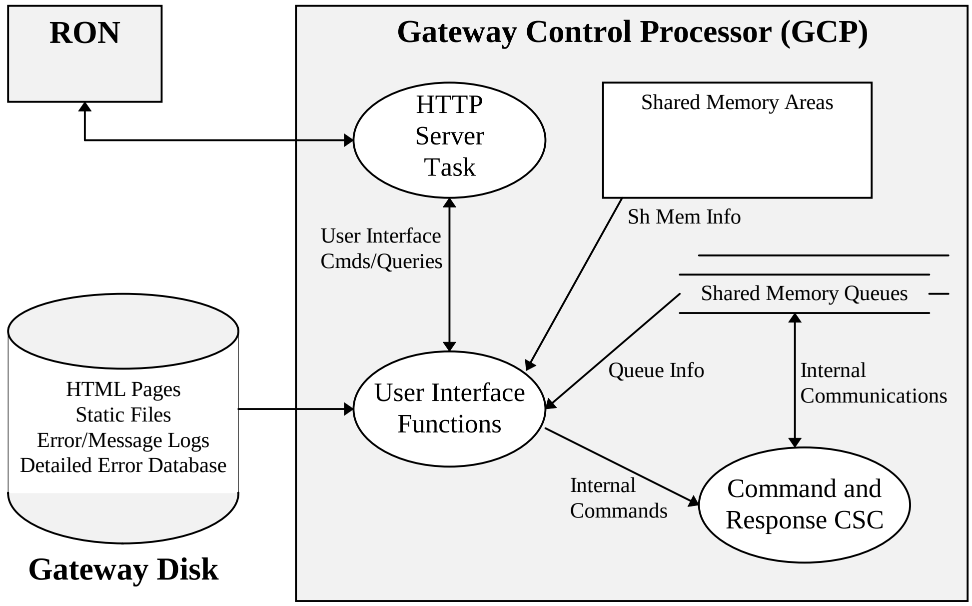
9.3 Gateway Maintenance User Interface CSC Design Specification

The Gateway Maintenance User Interface is primarily an HTTP Server running on the Gateway. Using a Web Browser from any workstation on the RON, a user will be able to gain Maintenence access to any Gateway. Each Gateway will have a base homepage which will contain information pertaining to the Common Gateway Services CSCI. This nature of this home page is depicted in 10.3.2.2. In addition, each Gateway will have a Gateway specific page depending on the number and type of FEPCs in the Gateway. On this page will be all information pertaining to that specific type of Gateway (e.g. PCM, GSE, etc.).

The Web Browsers on the RON machines will be communicating with the HTTP Server Task. This task will be supporting the Gateway's Home Page which will be read from the Gateway Disk. There are several buttons on the Home Page which can call different User Interface Functions. These functions are C functions which do the work to accomplish a request. Tha capabilities to read Gateway Disk files, read Shared Memory Message Queues, read Shared Memory Areas, and send Route Code 0 (internal) commands will all be executed using the User Interface Functions.

9.3.1 Gateway Maintenance User Interface CSC Detailed Data Flow

Common Gateway Services CSCI
Gateway Maintenance User Interface CSC

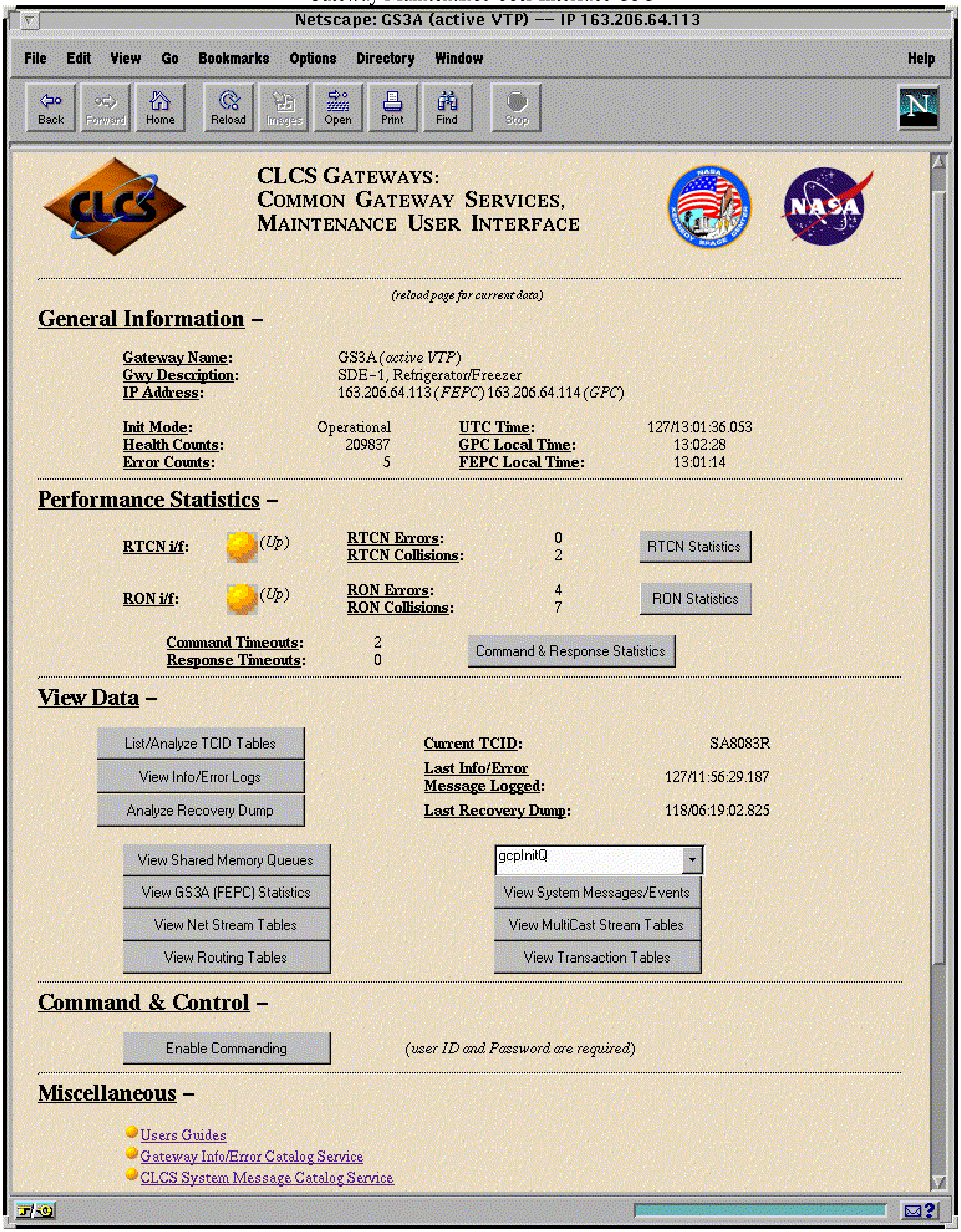


9.3.2 Gateway Maintenance User Interface CSC External Interfaces

9.3.2.1 Gateway Maintenance User Interface CSC Message Formats

None

9.3.2.2 Gateway Maintenance User Interface CSC Display Formats



9.3.2.3 Gateway Maintenance User Interface CSC Input Formats

None

9.3.2.4 Gateway Maintenance User Interface CSC Recorded Data

None

9.3.2.5 Gateway Maintenance User Interface CSC Printer Formats

None

9.3.2.6 Gateway Maintenance User Interface CSC Interprocess Communications

Gateway Maintenance User Interface internal interprocess communcation is using the GCP Services API CSC.

9.3.2.7 Gateway Maintenance User Interface CSC External Interface Calls

None

9.3.2.8 Gateway Maintenance User Interface CSC Table Formats

None

9.3.3 Gateway Maintenance User Interface CSC Test Plan

9.3.3.1 Environment

The Common Gateway Services CIT will be performed in conjunction with the GSE Gateway Services CIT. A development Gateway will be configured as a GSE Gateway. RTCN commands will be sent to the development Gateway. The action taken and the responses returned will be verified.

9.3.3.2 Test Tools

None

9.3.3.3 Test Cases

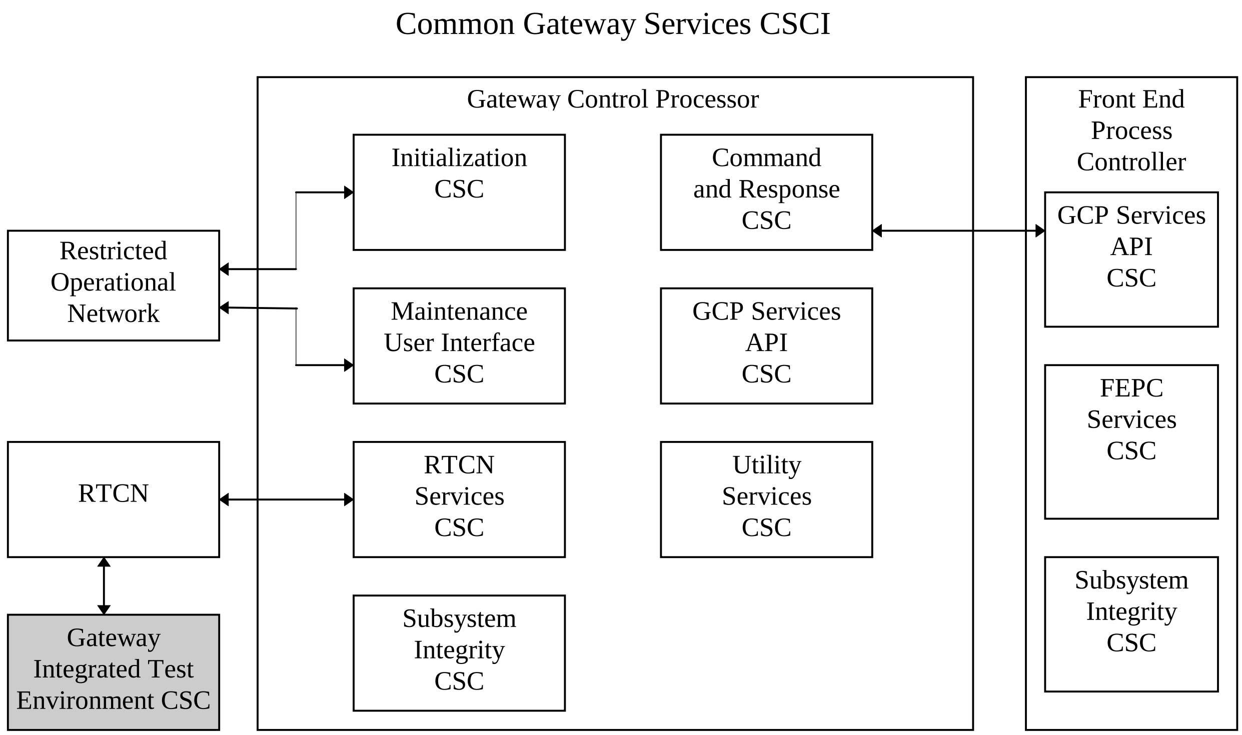
1. Verify functionality of the base Home Page.

10. Gateway Integrated Test Environment CSC

10.1 Gateway Integrated Test Environment CSC Introduction

10.1.1 Gateway Integrated Test Environment CSC Overview

The Gateway Integrated Test Environment CSC is responsible for testing the LDB, GSE, PCM and SIM Gateways. It is part of the Common Gateway Services CSCI and is and will be resident on the testing platform. The testing platform can be a SunOS or IRIX machine that has access to the RTCN.



10.1.2 Gateway Integrated Test Environment CSC Operational Description

The Gateway Integrated Test Environment CSC is a fully graphical integrated environment that will allow developers to test their Gateway systems verifying design requirements. The main GITE will consist of a Change Data Packet Analyzer, the Initialization Command Generator, and the Gateway Command Generator.

10.2 Gateway Integrated Test Environment CSC Specifications

10.2.1 Gateway Integrated Test Environment CSC Groundrules

- The GITE will support the following Gateways
  - Launch Data Bus Gateway
  - Ground Support Equipment Gateway
  - Pulse Code Modulation Gateway
  - Simulation Gateway

- All communication between the GITE and the Gateways will be via the RTCN.
- The Tool Command Language (TCL) COTS package is required to use the GITE.
- The GITE will use the System Services CSCI’s Multicast streams file for establishing an RTCN connection.

10.2.2 Gateway Integrated Test Environment CSC Functional Requirements

The Functional Requirements for the Gateway Integrated Test Environment CSC are arranged in the following major/minor functions:

1. GITE Configuration Options GUI
2. Initialization Command Generator
3. Gateway Command Generator
4. Change Data Analyzer
5. Function Designator Tracking Tool
6. HCI Applications Interface

10.2.2.1 GITE Configuration Options GUI

The GITE Configuration GUI will be the main configuration display for the GITE. It is here that the user can change any sort of configuration item in the GITE.

1. Connection to Gateways using the GITE shall be by Multicast Data Stream.
2. The GITE shall support all Muticast Data Streams available in the Multicast Streams Table.

10.2.2.2 Initialization Command Generator

The Initialization Command Generator is how the GITE emulates the Ops CM Server on the RTCN. A command that is chosen in the user interface will be encoded and sent over the Ops CM stream. After the command is sent the application will wait for a response.

1. All Initialization command and response packets shall be available for display at the user’s request.
2. Upon response of a non-zero completion code, the GITE shall alert the user and inform what type of code was received.
3. The GITE shall be capable of generating and responding to all Gateway Initialization commands.

10.2.2.3 Gateway Command Generator

The Gateway Command Generator will send the requested commands over the CCP data stream. The user interface will list all of the available commands and attributes associated with them for command creation.

1. Commands will be organized by Gateway type.
2. All Command Generator command and response packets shall be available for display at the user’s request.

10.2.2.4 Change Data Analyzer

The Change Data Analyzer will read change data packets from any Change Data Stream and display them in the GITE. The packet will be decoded so it can be easily understood.

1. When the Change Data Analyzer is activated, the GITE shall read RTCN Change Data Packets into a buffer.
2. The GITE Change Data Packet buffer size shall be configurable.
3. The GITE shall decode the Change Data packet’s header and body to be easily decipherable.
4. The GITE shall support the decoding of all Change Data body types.
5. The GITE shall provide Start and Pause capabilities while reading from the RTCN.
6. The GITE shall have the capability to clear the buffer so a fresh set of packets can be retrieved.

7. The GITE shall be capable of filtering out Change Data packets with no packet bodies.

10.2.2.5 Function Designator Tracking Tool

The Function Designator Tracking Tool will allow developers to select one or many FD's and monitor their current values as they change on the RTCN.

1. The GITE shall support real time monitoring of selected FD's via the Change Data stream.
2. The GITE shall support a maximum of five simultaneously tracked FD's.
3. The GITE shall support the historical graphing of FD's (Post-Thor).

10.2.2.6 LDB Applications Interface

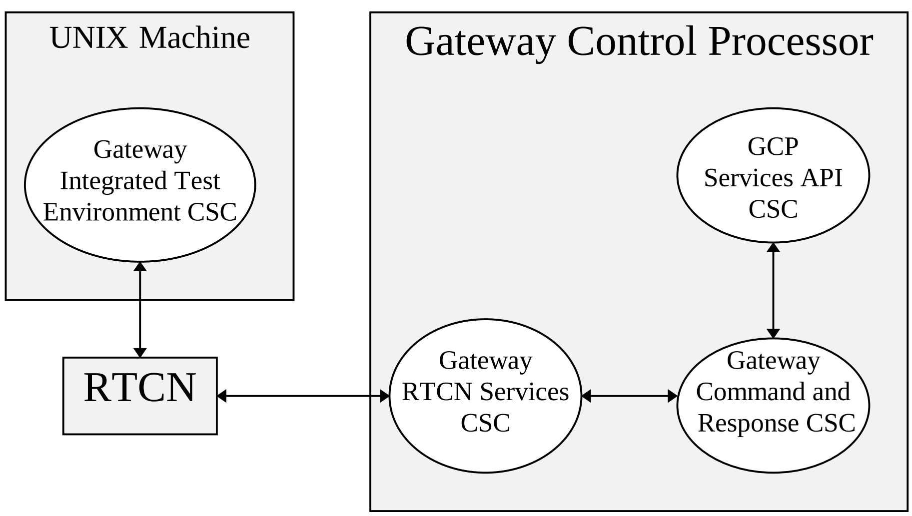
The LDB Applications Interface will be the connection to front-end applications from the GITE. Graphical LDB applications that are sending and receiving packets will be able to connect to the GITE so their command and response packets can be analyzed while still maintaining the full functionality of the application.

1. The GITE shall provide an API to for the LDB Applications Interface.
2. LDB applications shall interface to the GITE via C procedural calls. (send and read response)

10.2.3 Gateway Integrated Test Environment CSC Performance Requirements

No performance requirements have been identified for the Gateway Integrated Test Environment CSC for the Thor delivery.

10.2.4 Gateway Integrated Test Environment CSC Interfaces Data Flow Diagram



The communication to any Gateway using the GITE CSC will be over the RTCN.

10.3 Gateway Integrated Test Environment CSC Design Specification

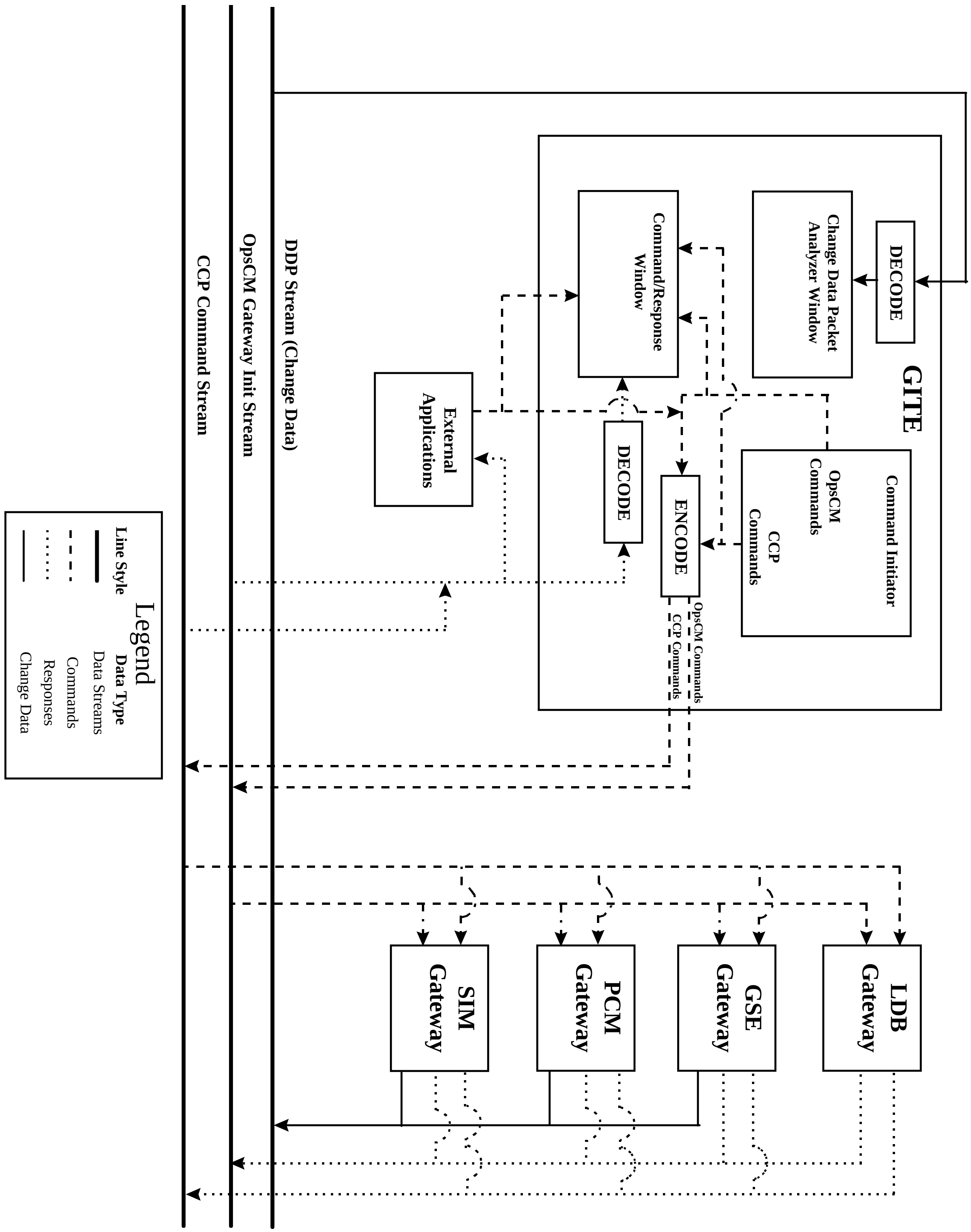
The Gateway Integrated Test Environment is responsible for fully testing the gateways being developed for the Checkout Launch and Control System. The CSC will provide multiple interfaces allowing developers to tests all aspects of the Gateways.

The data flow diagram shows how the GITE creates the commands. The Command Initiator dumps a decoded copy to the display window, encodes the command to the C-C format, and sends them on their appropriate stream. The destination gateway will receive the command from the streams, process it, and send out a response. This response is picked up by the GITE, decoded, and sent to the Command/Response display window.

The GSE, SIM, and PCM gateways produce Change Data that is put on the DDP stream. This change data is picked up by the Change Data Analyzer and decoded into a readable format, by the Change Data Analyzer

The External Applications are launched by the GITE and its commands will be routed through the GITE just as if the GITE initiated the command. Therefore all the command/response information from the external application will be logged in the GITE display window.

10.3.1 Gateway Integrated Test Environment CSC Detailed Data Flow



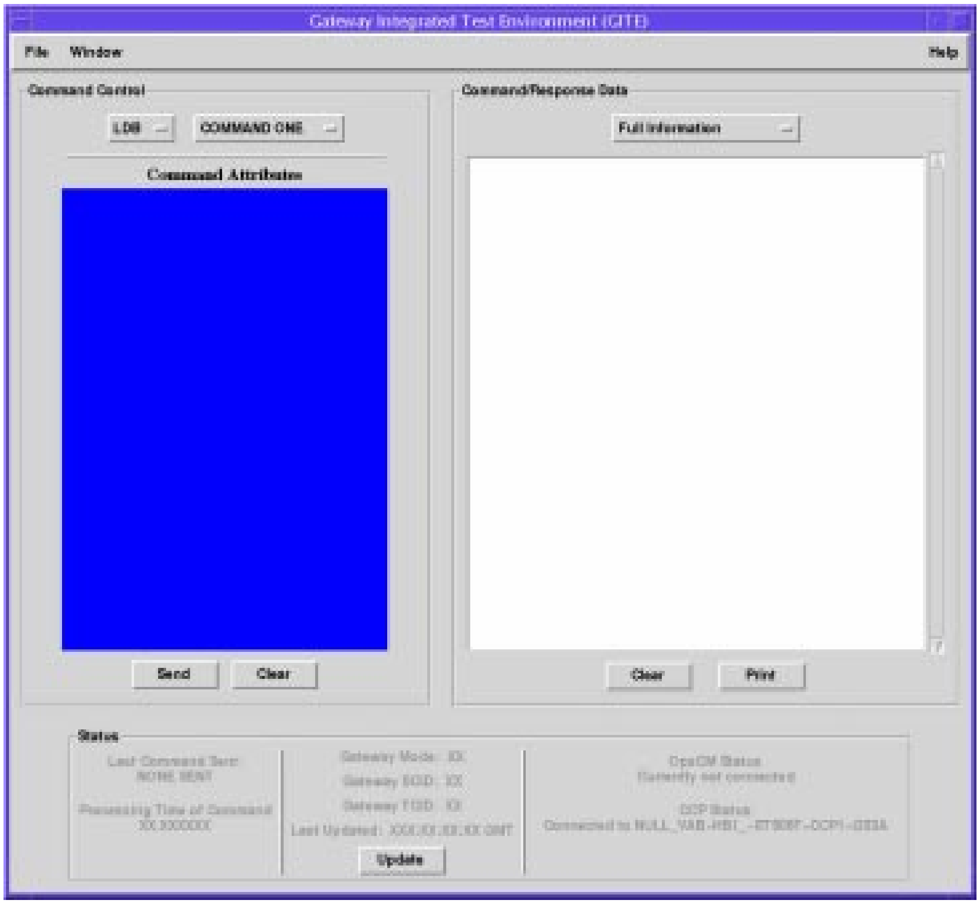
10.3.2 Gateway Integrated Test Environment CSC External Interfaces

10.3.2.1 Gateway Integrated Test Environment CSC Message Formats

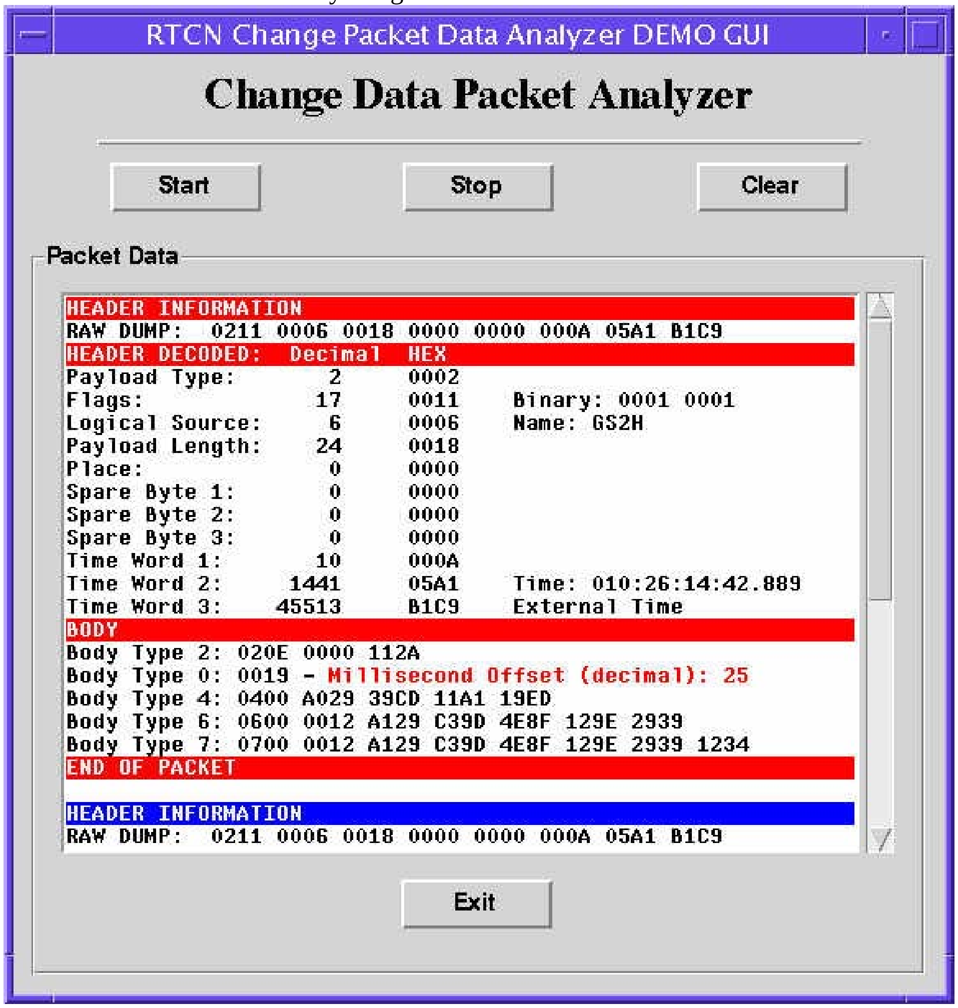
None

10.3.2.2 Gateway Integrated Test Environment CSC Display Formats

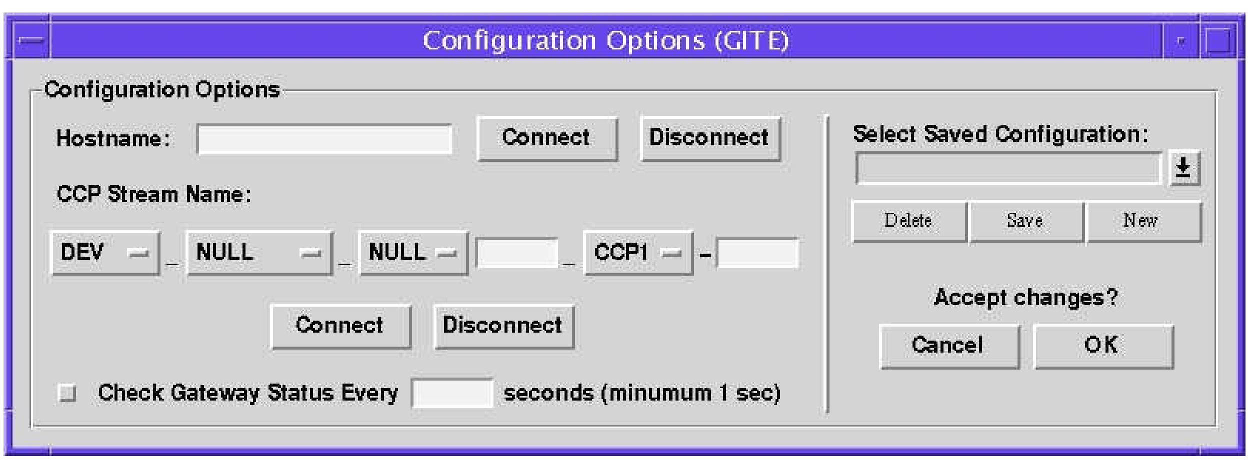
10.3.2.2.1 Command Initiator



10.3.2.2.2 Change Data Analyzer



10.3.2.2.3 Configuration Panel



10.3.2.3 Gateway Integrated Test Environment CSC Input Formats

None

10.3.2.4 Gateway Integrated Test Environment CSC Recorded Data

None

10.3.2.5 Gateway Integrated Test Environment CSC Printer Formats

10.3.2.5.1 Command and Response Window

The GITE has the option to send the contents of the command/response window to the printer. The format for this printout is listed here:

Printer Format for Commands (TEST DATA ONLY)

```
COMMAND HEADER INFORMATION
RAW DUMP:  0102 0304 0506 0708 0010 0087 0074 5E57 1112 1314
           1516 1718 191A 1B1C 1D1E 1F20 2122 2526 2728 2930
HEADER DECODED:        Decimal   HEX
Payload Type:                1     01
Flags:                       2     02   Binary: 0000 0010
Source RSYS ID:              3     03
Source Log. CPU ID:          4     04
Dest. 1 Log. RSYS ID:        5     05
Dest. 1 Log. CPU ID:         6     06
Dest. 2 Log. RSYS ID:        7     07
Dest. 2 Log. CPU ID:         8     08
Pyld. Length (Bytes):       16   0010
Time Word 1:               135   0087
Time Word 2:               116   0074
Time Word 3:             24151   5E57   Time: 135:02:07:06.327
Place:                      17     11
Single Spare byte:          18     12
Source Ref. Desig.:       4884   1314
Dest. 1 Desig.:           5398   1516
Dest. 2 Desig.:           5912   1718
Source Appl. ID:          6426   191A
Dest. 1 App. ID:          6940   1B1C
Dest. 2 App. ID:          7454   1D1E
Transaction ID:           7968   1F20
Routing Code:               33     21
Request ID:                 34     22
Array Spare Byte 1:         37     25
Array Spare Byte 2:         38     26
Array Spare Byte 3:         39     27
Array Spare Byte 4:         40     28
Array Spare Byte 5:         41     29
Array Spare Byte 6:         48     30
COMMAND BODY - APPLY ANALOG COMMAND
RAW DUMP:  0001 446F 7567 0076 7776 7574 7372 7170
BODY DECODED:       HEX             EU
FDID:             0001446F        ******
Requested Value:  448AE385        1111.109985
END OF COMMAND PACKET
```

Printer Format for Responses (TEST DATA ONLY)

```
RESPONSE HEADER INFORMATION
RAW DUMP:  0102 0304 0506 0708 0010 0087 0074 5E57 1112 1314
           1516 1718 191A 1B1C 1D1E 1F20 2122 0002 2526 2728
HEADER DECODED:        Decimal  HEX
Payload Type:              1       01
Flags:                     2       02   Binary: 0000 0010
Resp. RSYS ID:             3       03
Resp. Log. CPU ID:         4       04
Dest. 1 Log. RSYS ID:      5       05
Dest. 1 Log. CPU ID:       5       06
Dest. 2 Log. RSYS ID:      5       07
Dest. 2 Log. CPU ID:       5       08
Pyld. Length (Bytes):     16     0010
Time Word 1:             135     0087
Time Word 2:             116     0074
Time Word 3:           24151     5E57   Time: 135:02:07:06.327
Place:                    17       11
Single Spare byte:        18       12
Resp. Ref. Desig.:      4884     1314
Dest. 1 Desig.:         5398     1516
Dest. 2 Desig.:         5912     1718
Resp. Appl. ID:         6426     191A
Dest. 1 App. ID:        6940     1B1C
Dest. 2 App. ID:        7454     1D1E
Transaction ID:         7968     1F20
Trans. ID Respg. to:    8482     2122
Completion Code:           2     0002
Array Spare Byte 1:       37       25
Array Spare Byte 2:       38       26
Array Spare Byte 3:       39       27
Array Spare Byte 4:       40       28
BODY - APPLY ANALOG COMMAND
RAW DUMP:  447F DCCD 448A E385 0809 447F DCCD 448A
BODY DECODED:               HEX             EU
FDID:                      447FDCCD        ******
Requested Value:           448AE385        1111.109985
Received Value:            0809447F        0.000000
Transmitted Raw Counts:        DCCD        Dec: 56525
Received Raw Counts:           448A        Dec: 17546
Completion Code response: INVALID_FDID
END OF PACKET
```

10.3.2.5.2 Change Data Packet Analyzer Window

The GITE has the option to send the contents of the cahnge data analyzer window to the printer. The format for this printout is listed here:

```
HEADER INFORMATION
RAW DUMP:  0211 0006 0018 0000 0000 000A 05A1 B1C9
HEADER DECODED:  Decimal  HEX
Payload Type:       2     0002
Flags:             17     0011    Binary: 0001 0001
Logical Source:     6     0006    Name: GS2H
Payload Length:    24     0018
Place:              0     0000
```

```
Spare Byte 1:       0     0000
Spare Byte 2:       0     0000
Spare Byte 3:       0     0000
Time Word 1:       10     000A
Time Word 2:     1441     05A1        Time: 010:26:14:42.889
Time Word 3:    45513     B1C9    External Time
BODY
Body Type 2: 020E 0000 112A
Body Type 0: 0019 - Millisecond Offset (decimal): 25
Body Type 4: 0400 A029 39CD 11A1 19ED
Body Type 6: 0600 0012 A129 C39D 4E8F 129E 2939
Body Type 7: 0700 0012 A129 C39D 4E8F 129E 2939 1234
END OF PACKET
```

10.3.2.6 Gateway Integrated Test Environment CSC Interprocess Communications

The GITE has the capability to generate all GSE, PCM D/L, and LDB commands defined for Thor. These command formats are covered in the CSCI Design Documents of those three Gateway types.

10.3.2.7 Gateway Integrated Test Environment CSC External Interface Calls

The GITE has an external inteface for the External Applications that utilize the GITE for test purposes. There is only one C procedure call.

10.3.2.7.1 External Application Send Command

int gite_send_cmd (cmd, *resp_ptr)

Parameters:

cmd: The C-C command structure.

resp_ptr: A pointer to the location to store the resposne.

The Command and Response C-C structures are defined in the Thor Packet Payload Definition Document.

10.3.2.8 Gateway Integrated Test Environment CSC Table Formats

10.3.2.8.1 Reliable Messaging Multicast Table (as defined in Gateway RTCN Services CSC)

10.3.3 Gateway Integrated Test Environment CSC Test Plan

10.3.3.1 Environment

The Common Gateway Services CIT will be performed in conjunction with the GSE Gateway Services CIT. A development Gateway will be configured as a GSE Gateway. RTCN commands will be sent to the development Gateway. The action taken and the responses returned will be verified.

10.3.3.2 Test Tools

The Gateway Integrated Test Environment CSC is the Test Tool used to check out all other Common Gateway Services CSCs. The Test Data Generator Data Stream will be used to check out the RTCN Analyzer and FD Tracker.

10.3.3.3 Test Cases

1. Generating an RTCN C-C Command.
2. Attaching to an RTCN stream.